universität wien

DISSERTATION / DOCTORAL THESIS

Titel der Dissertation / Title of the Doctoral Thesis

Simulation von Material- und Informationsflüssen zur Analyse und Verbesserung von Instandhaltungsprozessen

verfasst von / submitted by

Mag. Christoph Stephan Prackwieser

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Doktor der Sozial- und Wirtschaftswissenschaften (Dr. rer. soc. oec.)

Wien, 2017 / Vienna 2017

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 084 175
Dissertationsgebiet lt. Studienblatt / field of study as it appears on the student record sheet:	Wirtschaftsinformatik
Betreut von / Supervisor:	o. Univ.-Prof. Dr. Dimitris Karagiannis

Inhaltsübersicht

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

1.1 Motivation

Wettbewerbsorientierte Supply Chains standardisieren ihre internen Prozesse um Skaleneffekte zu erzielen, müssen aber gleichzeitig flexibel genug bleiben, um auf plötzliche Nachfrageschwankungen und Störungen reagieren zu können. Es gilt, eine Balance zwischen Kosten, Lieferbereitschaft und Reaktionsfähigkeit zu finden. Sehr ähnliche Herausforderungen werden an Instandhaltungsmanager und an die von ihnen implementierten Prozesse und Verfahren gestellt (Prackwieser, 2013):

- im normalen Betriebszustand einer Produktionsanlage müssen deren Betriebskosten so gering wie möglich gehalten werden und
- im Fall von Betriebsstörungen, beispielsweise verursacht durch einen Anlagendefekt, muss eine Wiederherstellung der Betriebsfähigkeit der Anlage kurzfristig und kostengünstig erfolgen.

Dies bedeutet, dass während des Betriebs einer Anlage alle notwendigen instandhaltungsrelevanten Maßnahmen, wie Inspektionen oder Wartungstätigkeiten ressourcenschonend und mit möglichst geringer Auswirkung auf die Produktion durchgeführt werden müssen. Es ergibt sich daraus die Tendenz, dass Vor-Ort-Lagerbestände an Ersatzteilen reduziert werden und Instandhaltungstätigkeiten und damit einhergehendes Know-how mittels Dienstleistern externalisiert wird (Rasch, 2000). Tritt jedoch eine ungeplante Betriebsunterbrechung auf, muss die Instandhaltungs-Organisation in der Lage sein, schnell und flexibel auf die Störung reagieren zu können und alle Ressourcen, die zur Wiederherstellung des betriebsfähigen Zustand notwendig sind, kurzfristig bereitstellen können. Die Ausfallkosten werden dabei maßgeblich von der Dauer der Unterbrechung bestimmt. Ressourcen zur Wiederherstellung des Betriebszustands sind beispielsweise benötigte Informationen, Ersatzteile, Spezialisten und Werkzeuge (Pérès und Noyes, 2006).

Das zur Bereitstellung der Ressourcen und Durchführung aller Instandhaltungstätigkeiten notwendige Netzwerk an Informationsflüssen, Materialflüssen und Prozessschritten wird als Instandhaltungs- oder auch Maintenance Supply Chain (MSC) bezeichnet (Fabry und Schmitz-Urban, 2010). Sie reicht von Lieferanten der Ersatzteile und Dienstleistungen bis zum Abnehmer, der im Falle einer MSC eine instand zu haltenden Anlage ist.

Ein maßgebliche Erfolgsfaktor für die Effizienz einer Supply Chain im Allgemeinen und einer MSC im Speziellen ist die wirksame Koordination übergreifender logistischer Abläufe wie Material- und Informationsflüsse und die reibungsfreie Schnittstellengestaltung zwischen den

einzelnen, in die Supply Chain involvierten Organisationen (Fettke, 2007). Der Weg von Materialien durch die Supply Chain wird als Materialfluss bezeichnet (siehe Kapitel 2.3.5.1), ausgetauschte Informationen bilden den Informationsfluss. Viele der Transport- und Bearbeitungsschritte des Materialflusses werden durch Steuerungssignale des Informationsflusses ausgelöst. Diese Signale können wiederrum Resultat vorhergehender Materialbewegungen, abgearbeiteter Pläne oder externer Einflüsse sein. Je größer ein Supply Chain Netzwerk und je mehr materielle Teile in dessen logistischen Prozess involviert sind, desto komplexer wird dieses Zusammenspiel und desto wichtiger wird eine enge Verknüpfung und verzögerungsfreie Kopplung von Informations- und Materialflüssen. Das zeitliche Zusammenspiel und die gegenseitigen Abhängigkeiten von Informations- und Materialfluss wird in dieser Arbeit als Synchronisation (Lee, 2001) bezeichnet.

1.2 Zielsetzung der Arbeit und Forschungsfrage

Ziel dieser Arbeit ist es, eine praxisnahe Methode zu entwickeln, deren Anwendung die Synchronisation von Informations- und Materialflüssen in Instandhaltungs-Supply Chains transparent macht, den Anwender dabei unterstützt Verbesserungspotentiale zu identifizieren, diese zu bewerten, zu kommunizieren und umzusetzen.

Die Kopplung von Informations- und Materialflüsse wird in modernen Supply Chains durch aktuelle Technologien, wie RFID (Radio Frequency Identifikation) oder IoT (Internet of Things) unterstützt. Die Methode muss die Abbildung und Analyse dieser Konzepte unterstützen und eine Verfolgung einzelner Teile durch das System der Supply Chain ermöglichen.

Eine Literaturrecherche, siehe Kapitel 2.5.1f, hat ergeben, dass im Vergleich zu analytischen Verfahren, basierend auf mathematischen Modellen eine graphische Modellierung und Simulation zum einen flexibler in der Anwendung ist und zum anderen verständlicher für Praktiker ist. Dies entspricht auch der Erfahrung des Autors, deshalb behandelt diese Dissertation die folgende Forschungsfrage:

Wie ist eine graphische Modellierungsmethode zu gestalten, mit der das Zusammenspiel von Material- und Informationsflüssen einer (Instandhaltungs-) Supply Chain dokumentiert, analysiert und verbessert werden kann?

Die in dieser Arbeit entwickelte Methode ist unter dem Namen *SIMchronization* auf der Metamodellierungsplattform ADOxx prototypisch umgesetzt worden und wird mittels der Plattform Open Models Laboratory (OMiLAB) der wissenschaftlichen Community zur Verfügung gestellt.

1.3 Literaturüberblick

Alabdulkarim et al. (Alabdulkarim et al., 2011) stellen in ihrer Arbeit fest, dass die Modellierung von Instandhaltungsoperationen komplex ist und noch nicht so weit entwickelt und erforscht ist, wie beispielsweise die Modellierung von Produktionssystemen. Als Begründung dafür sehen sie, dass hier viele Sub-Systeme miteinander „in a complex fashion“ interagieren. Deshalb werden die einzelnen Teil-Systeme, wie beispielsweise Produktion, Instandhaltungsmaßnahmen oder Ersatzteillagerhaltung, üblicherweise isoliert voneinander betrachtet. In Duffuaa et al. (Duffuaa et al., 2001) und (Andijani und Duffuaa, 2002) wird ein generisches konzeptuelles Modell vorgestellt, das zum Ziel hat, die verschiedenen Teil-Systeme zu integrieren und gemeinsam zu betrachten. Auch wenn diese Literaturbeiträge mathematische Modellierung besprechen, so gilt ähnliches für die graphische Modellierung. Der in dieser Arbeit verfolgte Ansatz hat zum Ziel die oben genannten Sub-Systeme in einem Modell abzubilden und zu analysieren.

Viele Arbeiten präsentieren Ansätze zur Entscheidungsunterstützung für spezifische Fragestellungen der Instandhaltungsoptimierung und Konfigurationen von Maintenance Supply Chains. Beispielsweise werden zur Optimierung des Lagermanagements ein- oder mehrdimensionale Klassifikationssysteme zur Ermittlung der Kritikalität von Ersatzteilen vorgestellt (Kennedy et al., 2002). Zur Selektion einer passenden Instandhaltungspolitik, welche die Kosten minimiert und die Verfügbarkeit und Zuverlässigkeit erhöht, wird mathematische Modellierung mit analytischer Optimierung vorgeschlagen (Van Horenbeek et al., 2010). Allen Verfahren, die auf mathematischer Modellierung beruhen, ist jedoch gemein, dass sie jeweils nur für die Lösung einer sehr spezifischen Problemsituation entwickelt worden sind und darüber hinaus dem Anwender nur wenige Freiheitsgrade bei der Adaption des Modells an individuelle Problemstellungen lassen (Van Horenbeek et al., 2010) und (Prackwieser, 2013). Laut Wang (Wang, 2002) erfolgt die Optimierung der meisten Modelle nach genau einem Zielkriterium. Er argumentiert, dass dies unzureichend ist, um reale Fragestellungen der Instandhaltung abzubilden. Dekker (Dekker, 1996) und Van Horenbeek (Van Horenbeek et al., 2010) argumentieren des Weiteren, dass die existierenden mathematischen Modelle für Instandhaltungspraktiker nur schwer zu verstehen sind und darüber hinaus die Anwendbarkeit dieser Verfahren durch fehlende Flexibilität leidet.

Betrachtet man die Literatur in Bezug auf graphische Modellierung von Supply Chains und im Speziellen von Maintenance Supply Chains, so fällt auf, dass viele Arbeiten die Maintenance Supply Chain mittels klassischen Geschäftsprozesssprachen abbilden. Der Kontrollfluss eines Prozesses gibt dabei die sequentielle Reihenfolge der Abarbeitung von Tätigkeiten vor. Je nach verwendeter Methode werden Hinweise zum Informationsfluss, beispielsweise mittels benötigter Inputs und generierter Outputs gegeben.

Beispiele, die diesem Ansatz folgen, sind:

- Das Project „Maintenance Supply Chain Optimisation“[1] (Meier und Klimek, 2008), (Klimek, 2011) verfolgte den Ansatz, mittels einer medienbruchfreien Kommunikation den Informationsfluss zu verbessern und folglich die Durchlaufzeiten der Ersatzteillieferkette zu verringern. Das Projekt entwickelte eine offene IT-Service-Plattform über welche diese Kommunikation erfolgt. Um die Vorteile des Ansatzes zu zeigen, wurden Maintenance Supply Chains mittels einer Geschäftsprozessmodellierungsmethode abgebildet und deren Durchlaufzeitverhalten simuliert. Materialflüsse wurden in dieser Arbeit nicht berücksichtigt.
- Das EU-Forschungsprojekt Projekt ComVantage[2], auf das in Kapitel 0 näher eingegangen wird, arbeitete an einer IT-Architektur zur Ausführung von Geschäftsprozessen in kollaborativen Unternehmensnetzwerken. Dabei werden mobile Applikationen und „Linked Data“ genutzt, um beispielsweise das Anwendungsgebiet der mobilen Instandhaltung (Buchmann, 2014) zu unterstützen. Ein betrachtetes Szenario dabei ist die Kommunikation von Sensoren, die als Teil des Internet of Things (IoT) mit dem Instandhaltungskoordinator kommunizieren. Im Rahmen des Projekts wurde eine hybride Modellierungsmethode zur Abbildung von Geschäftsprozessen der betrieblichen Instandhaltung entwickelt. Diese Methode umfasst Konstrukte, mit denen Anforderungen an Ressourcen und IT-Services spezifiziert werden können. Auch wenn ComVantage dieselbe Domäne, wie die, in dieser Arbeit entwickelte Methode unterstützt, unterscheidet sich der verfolgte Ansatz zur Abbildung und Simulation der Prozesse stark. So können ComVantage-Modelle nicht hinsichtlich der durch individuelle Objekte des Material- oder Informationsflusses ausgelösten Aktionen und Reaktionen hin analysiert werden und eine Analyse von Lagerbeständen ist nicht möglich.
- Sarac et al. (Sarac et al., 2015) stellen einen Ansatz zur Simulation des Einflusses von RFID-Technologie (siehe Kapitel 2.3.7.2) auf Supply Chains vor. Sie verwenden eine kontrollflussorientierte Modellierungssprache zur Abbildung des Simulationsmodells. Auch Amini et al. (Amini et al., 2007) und Al-Safedi und Al-Sulaiman (Al-Safadi und Al-Sulaiman, 2011) wenden diesen Ansatz zur Evaluation der RFID-Technologie an. Joshi (Joshi, 2000) und Lee et al. (Lee et al., 2005) zeigen die quantitativen Auswirkungen der erhöhten Informationsverfügbarkeit durch den Einsatz von RFID mittels Simulation. Beispielsweise zeigen Lee et al. dass ein Lieferant mittels RFID sowohl den Lagerbestand verringern, als auch die Lieferfähigkeit verbessern kann.

[1] Siehe http://data.fir.de/projektseiten/msco/ Zugriff am 26.03.2017

[2] Siehe http://comvantage.eu Zugriff am 25.03.2017

Andere Methoden beschreiben den Fluss von Material durch die Wertschöpfungskette. Dies ist ein Ansatz der auch von vielen kommerziellen Simulationstools verfolgt wird. Einige Ansätze bieten die Möglichkeit sowohl Informations- als auch Materialfluss abzubilden und zu analysieren:

- Persson et al. (Persson und Araldi, 2009), (Persson, 2011) und (Persson et al., 2012) stellen mit dem „SCOR Template“ einen Modellierungs- und Simulationsansatz vor mit dem SCOR-Modelle analysiert werden können. Das Supply Chain Operations Reference Model (SCOR) wird im Kapitel 2.3.8 dieser Arbeit vorgestellt. Das von ihnen präsentierte Verfahren erlaubt die Darstellung von Material- und Informationsflüssen, wenngleich in der Version 3 der für den Informationsfluss sehr wichtige „Plan“ Prozess noch nicht enthalten ist (Persson et al., 2012, S. 3831). Die Simulation ist mengenbasiert, eine Betrachtung individueller Teile und Informationsweitergaben ist nicht möglich. Aussagen über die Synchronisation von Material- und Informationsfluss können daher mit dieser Methode nicht getroffen werden.
- IBM SmartSCOR (Jin et al., 2006) bietet einen breiten Funktionalitätsumfang, um eine Supply Chain auf strategischer, taktischer und operativer Ebene zu analysieren. Es können Material- und Informationsflüsse abgebildet werden, allerdings ist auch hier die Simulation mengenbasiert, die Bewegung einzelner Teile durch das System kann nicht analysiert werden.
- Umeda et al. (Umeda und Jain, 2004) entwickelte über mehrere Jahre hinweg das Integrated Supply Chain Simulation System (ISSS). Die Modellierung unterstützt Material- und Informationsflussabbildung und folgt damit einen ähnlichen Ansatz wie diese Arbeit, das Unternehmen wird aber breiter betrachtet und das Prozess- und Rollenverständnis unterscheidet sich von dem, in der vorliegenden Arbeit verwendetem. Die zugrundeliegende Simulation ist ebenfalls mengenbasiert, eine Abbildung der RFID-Technologie ist damit nicht möglich.
- Zülch und Stock (Zülch und Stock, 2010) nutzen Wertstromdesign um einen Produktionsprozess in der Baustoffindustrie zu modellieren und analysieren. Die verwendete Modellierungsmethode unterstützt die Abbildung von Informations- und Materialflüssen. Sie wenden auf die statischen Wertstrommodelle eine Simulation an, da sie darin eine Möglichkeit sehen, die Planung von Prozessen besser zu unterstützen und darüber hinaus deren dynamische Leistungsfähigkeit analysieren und verbessern zu können. Auch das hier angewendete Verfahren ist mengenbasiert, es bietet aber eine Möglichkeit an, Störungen zu simulieren.
- Stäblein et al. (Stäblein et al. 2007) stellen den Supply Net Simulator (SNS) vor. Das Werkzeug wendet Elemente künstlicher Intelligenz, wie agentenbasierte Simulation, an, um Supply Chain Netzwerke zu simulieren und zu optimieren. Die Agenten optimieren dabei ihr Verhalten im System im Sinne eines Advanced Planning Systems.

Im Unterschied zum Ansatz von Almeder und Preusser (Almeder et al., 2009) und (Almeder und Preusser, 2007a) interagieren dabei die Simulations- und Optimierungskomponente nicht.

- Parlings et al. (Parlings und Motta, 2015) und (Parlings et al., 2016) stellen das Resultat des Forschungsprojekts „Supply Chain Design“ vor, in dem ein generalisierter Planungsworkflow für Supply Chains entwickelt wurde. Das konzeptionelle Modell der technischen Lösung sieht drei On-demand Services vor, den Modellierungs-, Simulations- und Reportingdienst. Besonders interessant am vorgestellten Ansatz ist, dass das Supply Chain Modell in eine Makro- und Mikro-Ebene unterteilt wird. Für die Makro-Ebene wurde eine fachliche domänenspezifische Modellierungssprache entwickelt, die vorkonfigurierten Modellbausteine anbietet, in denen die fachlichen Modellparameter gepflegt werden können. Die Mikro-Ebene transformiert diese fachlichen Modelle in ein, für das Simulationstool verständliches Format. Als Simulationstool wird OTD-NET (Wagenitz, 2007) verwendet. Nach der Simulation werden die Ergebnisse in die fachliche Modellebene zurück-aggregiert. Da die Transformation theoretisch auf andere Simulationstools angepasst werden kann, ist dieser Ansatz auch für die Erstellung des konzeptionellen Modells dieser Arbeit von Interesse. Allerdings bietet auch dieses Verfahren keine teileindividuellen Analysen an.

Aus der Literaturstudie ist ersichtlich, dass relativ viele Ansätze und Verfahren existieren, um Supply Chains zu modellieren und zu analysieren. Die Beschreibung spezifischer Modellierungsmethoden für Maintenance Supply Chains konzentriert sich aktuell noch auf mathematische Modellierung und analytische Auswertungsverfahren. Es konnte im Rahmen der Literaturrecherche kein Ansatz gefunden werden, der die in der Forschungsfrage aufgeworfene Anforderung nach einer Analysierbarkeit des Zusammenspiels von Informations- und Materialfluss auf Basis von individuellen Informationsobjekten und physischen Teilen erfüllt.

1.4 Aufbau der Arbeit

Nach der Einleitung, Beschreibung der Zielsetzung der Arbeit und Literaturrecherche wird in Kapitel 2 „Grundlagen“ das theoretische Grundgerüst zur Domäne Maintenance Supply Chain und dem Lösungskonzept der graphischen Modellierung gelegt. Folgend dem OMiLAB Ansatz, beschrieben in Kapitel 2.2.3 „Der OMiLAB Life Cycle“, werden in Kapitel 3 aus den gewonnenen Erkenntnissen Anforderungen strukturiert abgeleitet und beschrieben. Aus den Anforderungen wird ein konzeptionelles Modell der Modellierungsmethode in Kapitel 4 „Konzeptionelles Modell“ entwickelt, das in Kapitel 5 „Metamodell“ mit dem

Design des Metamodells verfeinert wird. Die formale Ausgestaltung des Metamodells findet sich in Kapitel 6 „Beschreibung des Metamodells in FDMM“.

Einen Überblick über die prototypische Implementierung auf der Plattform ADOxx bietet Kapitel 7 „Implementierung“. Die Evaluation der Methode anhand eines Abgleichs der Anforderung mit dem entwickelten Metamodell, beziehungsweise der Funktionalität des Prototyps, wird in Kapitel 8 „Evaluation“ beschrieben. Kapitel 9 „Modellierungsverfahren“ bietet einen Ausblick auf die Anwendung der Methode.

2. Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen zu der, dieser Arbeit zugrundeliegenden Vorgehensweise zur Entwicklung und Implementierung von Modellierungsmethoden dargestellt, Auswertungsmechanismen besprochen und in die Domäne der Maintenance Supply Chains eingeführt.

2.1 Der Modellbegriff

Auch wenn in verschiedenen Wissenschaftsdisziplinen der Modell-Begriff unterschiedlich verstanden und damit definiert wird (Schütte, 1999), so lassen sich doch, speziell, wenn man sich auf betriebswirtschaftliche Modelle konzentriert, bestimmte Gemeinsamkeiten erkennen[3]. Das Modelverständnis das dieser Arbeit zu Grunde liegt, folgt dem von Karagiannis et al. (Karagiannis et al., 2007, S. 15) wonach Modelle einen Ausschnitt der Realität (Lehner et al., 1995, S. 27) oder einer Vision abbilden (Whitten et al., 2004). Sie werden erstellt, um einen bestimmten Zweck zu erfüllen und ein bestimmtes Ziel zu erreichen (Object Management Group (OMG), 2014), (Bézivin und Gerbé, 2001). In der Wirtschaftsinformatik (Thomas, 2005) werden immaterielle Modelle erstellt (Becker et al., 1995), deren Zweck es ist, den Gegenstandsbereich (das Objektsystem) zu erklären oder zu gestalten (Junginger, 2001, S. 32).

Gemäß Karagiannis et al. (Karagiannis et al., 2007, S. 17) sind Modelle „knowledge that can be operationalized" und können folgendermaßen genutzt werden:

- um als ein präskriptives Modell eine Vision zu spezifizieren.
- um eine semi-automatische Software-Implementation zu unterstützen, die zu einer softwaretechnischen Umsetzung oder Ausführung des Modells führt. Diese Arbeit sieht auch die Simulation eines Modells als eine Form der Modellausführung an.
- um als ein deskriptives Modell einen Zustand eines Objektsystems zu dokumentieren und somit beispielsweise Qualitätsmanagement oder Knowledge Management zu unterstützen (Karagiannis et al., 2000).
- um den aktuellen Zustand eines Systems zu erklären und mittels Analysen mögliche Strategien zu identifizieren, wie bestimmte Zielkriterien verbessert werden können (Kurpjuweit und Winter, 2007).

[3] Zur Geschichte des Modellbegriffs verweist Schütte (Schütte, 1999) auf (Müller, 1983, S. 23ff).

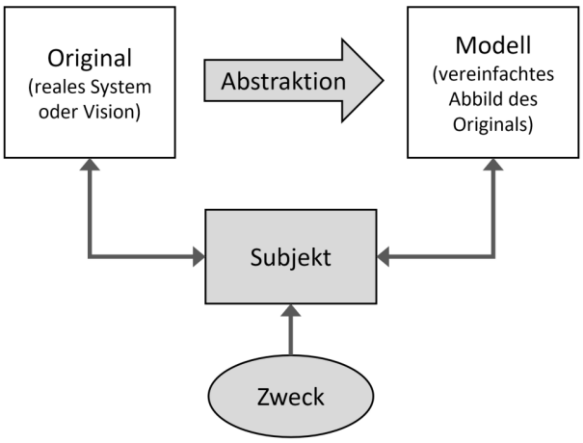


Abbildung 1: Modellierung nach Stachowiak (Stachowiak, 1974); mit Anpassungen übernommen aus (Mebes, 2008)

Jeder der aufgeführten Nutzungszwecke eines Modells hat unterschiedliche Anforderungen an die Struktur und den Inhalt eines Modells[4]. Der Modellersteller oder Modellnutzer wird von Stachowiak (Stachowiak, 1974) als Subjekt bezeichnet, siehe Abbildung 1. Die Tätigkeit der Modellierung ist immer mit Abstraktion verbunden, bei der, für das Modellierungsziel oder den Kontext nicht relevanten Aspekte vom Modellierer bewusst weggelassen werden[5]. Somit kann die Qualität eines Modells für den einen Nutzer hervorragend sein, während ein anderer Nutzer das Modell als unbrauchbar oder gar falsch ansieht (Karagiannis et al., 2016a).

Als Domäne, oder „system under study" (Seidewitz, 2003) wird der im Modell abgebildete Ausschnitt der Realität bezeichnet. Die vorliegende Arbeit ist in der Domäne des Supply Chain Managements angesiedelt und um in diesem breiten Anwendungsbereich konkrete Anforderungen für eine Analysemethode zu identifizieren, werden dabei im Speziellen instandhaltungsbezogene Supply Chains betrachtet.

2.1.1 Die Definition von Modellierungsmethoden

Die Modellierungssprache dient zur Explizitmachung eines Modells. Zur Abbildung graphischer Modelle bedarf es demgemäß diagrammbasierter Modellierungssprachen. Karagiannis und Kühn (Karagiannis und Kühn, 2002) entwickelten das Generic Modelling Method Specification Framework, welches die wesentlichen Komponenten einer Modellierungsmethode definiert und diese untereinander in Beziehung setzt. Demnach

[4] Eine interessante Diskussion zum Thema Performanz von Modellen und Detaillierungsgrad findet sich in Brooks und Tobias (Brooks und Tobias, 1996)

[5] Der Verein Deutscher Ingenieure drückt den Grad der möglichen Abstraktion in seiner Modell Definition folgendermaßen aus: „Ein Modell … unterscheidet sich hinsichtlich der untersuchungsrelevanten Eigenschaften nur innerhalb eines vom Untersuchungsziel abhängigen Toleranzrahmens vom Vorbild" (VDI, 2014)

werden Modellierungssprachen durch ihre graphische Notation, welche die grammatikdefinierende Syntax visualisiert, und ihre Semantik bestimmt, siehe Abbildung 2.

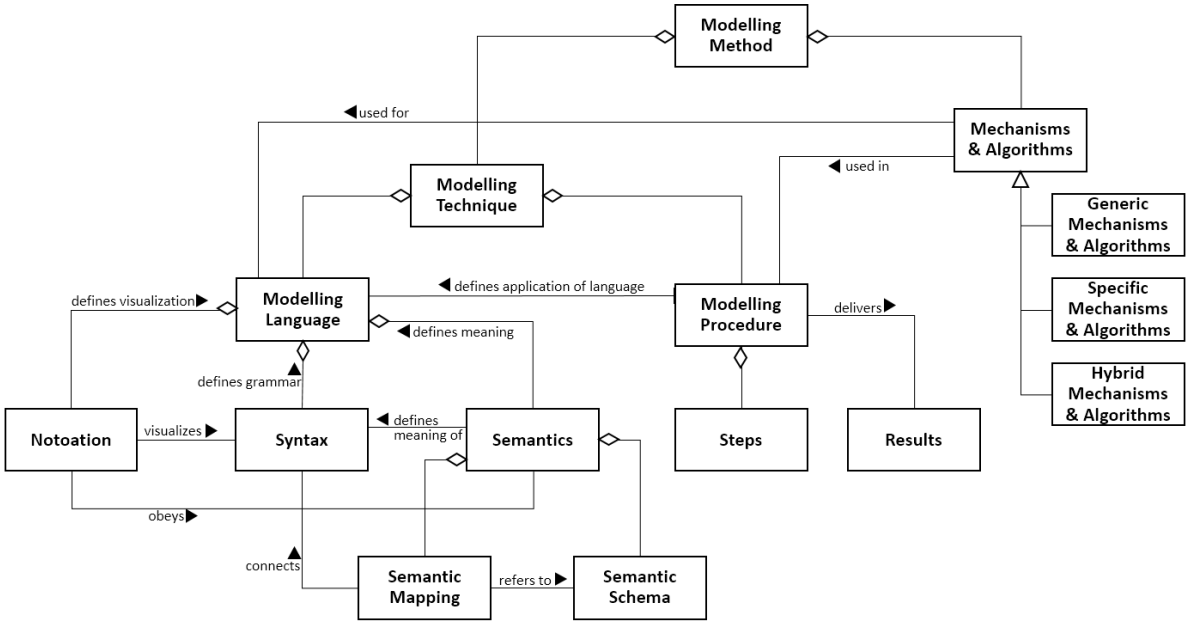


Abbildung 2: Komponenten einer Modellierungsmethode (Karagiannis und Kühn, 2002)

Ein oder mehrere Modellierungssprachen in Kombination mit einer zielgerichteten Handlungsanweisung zur Erstellung und Nutzung der Modelle - und nicht zuletzt Erzielung der gewünschten Resultate - nennen sie Modellierungstechnik. Um Modellierung und Nutzung der Modelle zu unterstützen, kann eine Modellierungsmethode darüber hinaus Algorithmen und Mechanismen anbieten. Diese können einerseits auf die, mittels der Modellierungssprache erstellten Modellen angewendet werden, um spezielle Auswertungen (Prackwieser et al., 2013) oder beispielsweise Modelltransformationen zu unterstützen. Andererseits können sie die Modellerstellung unterstützen, indem sie zum Beispiel sicherstellen, dass die vorgegebene Modellierungsprozedur eingehalten wird. Karagiannis und Kühn klassifizieren Algorithmen und Mechanismen des Weiteren in generische, spezifische und hybride Typen. Im Gegensatz zum spezifischen Typ können generische Algorithmen und Mechanismen auf beliebige Modellierungssprachen angewendet werden. Hybride Algorithmen und Mechanismen können zwar ebenfalls auf unterschiedliche Modellierungssprachen angewendet werden, bedürfen dafür aber einer Konfiguration. In Prackwieser et al. (Prackwieser et al., 2014) wird ein hybrider Simulationsalgorithmus beschrieben, der eines semantischen Mappings bedarf, um auf verschiedenen Geschäftsprozess-Modellierungssprachen angewendet zu werden.

Es gibt in der Literatur unterschiedliche Auffassungen zu den Bestandteilen von Modellierungsmethoden. Für Heym (Heym, 1993) und Gutzwiller (Gutzwiller, 1994) zählen zu den wesentlichen Komponenten einer Methode Aktivitäten, Rollen, Techniken, Ergebnisse

und das Metamodell. Für Holten (Holten, 2001) besteht eine Modellierungstechnik zwar ebenfalls aus einer Sprache und Handlungsanweisungen, aber beide Ansätze betrachten Mechanismen und Algorithmen, die auf die erstellten Modelle angewendet werden, nicht als zentralen Bestandteil einer Methode. Für die, in dieser Arbeit konzeptionierte Methode SIMchronization ist jedoch neben der Modellierungssprache und dem Vorgehensmodell, der zur Analyse verwendete Simulations- und Animationsalgorithmus von gleichrangiger Bedeutung (Prackwieser, 2013). Deshalb wird der Ansatz von Karagiannis und Kühn, der alle drei Komponenten, also Modellierungssprache, das Vorgehensmodell und Mechanismen und Algorithmen umfasst, für diese Arbeit als Rahmenwerk herangezogen. Dieses Framework unterstützt darüber hinaus die Operationalisierung der Modellierungsmethode auf Basis einer metamodellierungsfähigen Plattform. Dies ist für die, zur Evaluierung der Methode erforderliche prototypische Implementierung hilfreich.

2.1.2 Generische vs. Domänenspezifische Modellierungssprachen

Bevor ein Modellierer ein Original durch Abstrahierung in einem Modell abbilden kann, muss geklärt werden, in welcher Modellierungssprache dies am effizientesten erfolgt. Zwar werden in dieser Arbeit ausschließlich graphische Modellierungssprachen betrachtet, aber schon unter diesen steht eine Vielzahl von alternativen Sprachen oder Spracherweiterungen zur Verfügung. Auch wenn eine klare Abgrenzung nicht möglich ist, so werden Modellierungssprachen typischerweise dahingehend klassifiziert, wie spezifisch sie an den Anwendungsfall und -zweck, die Branche oder zum Beispiel ein IT-Ausführungssysteme angepasst sind. Man spricht in diesem Zusammenhang von generischen[6] oder domänenspezifischen[7] Sprachen. Beispiele für generische Sprachen sind Unified Modeling Language (UML) (OMG, 2015), Petri Netze, Ereignisgesteuerte Prozessketten (EPK), Business Process Modeling Notation (BPMN) oder ADONIS BPMS die jeweils in vielen unterschiedliche Domänen eingesetzt werden können. Methoden, wie MELCA (Hawryszkiewycz und Prackwieser, 2016) und PROMOTE (Karagiannis et al., 2000) sind zwar branchenübergreifend einsetzbar, aber speziell für die Einsatzgebiete des Design Thinking, im Falle von MELCA, und der Modellierung von Wissensprozessen, im Falle von PROMOTE, entwickelt worden. Karagiannis und Visic (Karagiannis und Visic, 2011) listen als Beispiele für GPLs auch Programmiersprachen wie Java und C# auf und erwähnen SQL, HTML und Excel als Beispiele für DSLs.

Um Wiederverwendbarkeit zu ermöglichen, enthalten diese Sprachen keine oder nur sehr wenig domänenspezifische Semantik und Konzepte. Domänenspezifische

[6] Auch bezeichnet als General Purpose Language (GPL)

[7] Auch bezeichnet als Domain Specific Language (DSL)

Modellierungssprachen hingegen beinhalten branchen- oder anwendungsnahe Konzepte, um die domänenspezifischen Besonderheiten effektiv und effizient abbilden zu können, dies schränkt jedoch ihre universelle Benutzung ein (Kirchner, 2007, S. 16).

2.2 Metamodellierung

Laut (Strahringer, 1996) und (Strahringer, 2016) ist ein Metamodell ein Beschreibungsmodell der Sprache eines anderen Modells, die auch als Objektsprache bezeichnet wird. Diese sogenannte sprachbasierte Auffassung (Atkinson und Kühne, 2003) der Metamodellierung bildet in einer Metasprache die Objektsprache ab. Dies kann rekursiv erfolgen, das Metametamodell (Meta²-Modell) hat dann das Metamodell zum Gegenstand und definiert eine Meta²-Sprache. Prinzipiell kann diese „Metaisierung" beliebig oft durchgeführt werden, aber mit Blick auf eine noch sinnvolle und ausdrucksstarke Abstraktionsebene wird sie üblicherweise auf vier Ebenen beschränkt (Karagiannis und Kühn, 2002), (Karagiannis und Höfferer, 2006). Abbildung 3 zeigt die, auch in ADOxx (siehe Kapitel 2.2.1) verwendeten Metamodell-Ebenen.

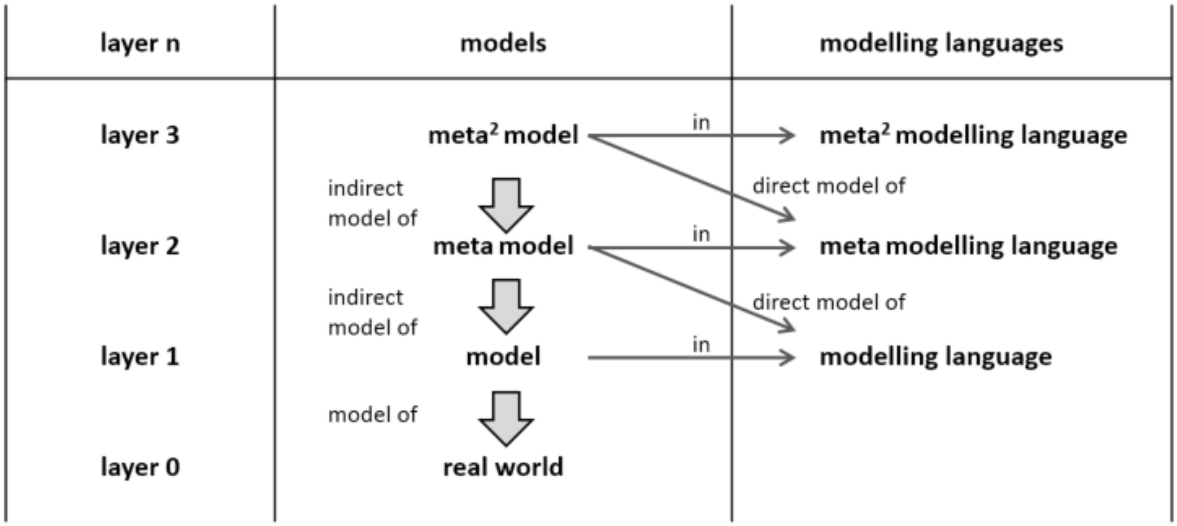


Abbildung 3: Metamodell-Ebenen basierend auf (Strahringer, 1996, S. 24) und (Karagiannis und Kühn, 2002)

Hinkelmann et al. (Hinkelmann et al., 2015) listen einige bekannte Metamodellierungssprachen basierend auf einer Arbeit von Kern et al. (Kern et al., 2011) auf:

- Ecore basierend auf der Eclipse Plattform (Budinsky et al., 2003)
- GOPRR basierend auf der MetaEdit+ Plattform (Kelly et al., 2005)
- MS DSL Tools auf Microsoft Visio (Cook et al., 2007)

Als weiteres Beispiel kann ADOxx genannt werden, das im folgenden Kapitel beschrieben wird und in dieser Arbeit Verwendung findet. ADOxx basiert auf der gleichnamigen Plattform ADOxx[8] und wird über OMiLAB der Community zur Verfügung gestellt.

2.2.1 ADOxx

ADOxx[9] ist eine Metamodellierungssprache und gleichzeitig der Name einer Metamodellierungsplattform in und mit der diagrammhafte Modellierungssprachen spezifiziert werden können. Die Plattform integriert die entwickelte Modellierungssprache automatisch in einen graphischen Modellierungseditor und stellt dieses Paket den Modellerstellern und Modellnutzern zur Verfügung (Fill et al., 2013b).

Der ADOxx zugrundeliegende Metamodellierungsansatz wird seit nunmehr über 20 Jahren (Junginger et al., 2000) sowohl in der Wissenschaft als auch in der unternehmerischen Praxis von einer Vielzahl von internationalen Projekten und Organisationen erfolgreich genutzt und erprobt. Er wird stetig, gemäß neuen wissenschaftlichen Erkenntnissen, praxisbasierten Anforderungen und Innovationen weiterentwickelt. ADOxx wird der wissenschaftlichen Community für Forschungsprojekte mittels der virtuellen und physischen Plattform Open Models Laboratory (OMiLAB) zur Verfügung[10] gestellt. Karagiannis et al. (Karagiannis et al., 2016b) enthält eine Auflistung und Beschreibung einiger dieser Forschungsprojekte. Kommerzielle Anwender können die auf ADOxx basierenden Produkte ADONIS, ADOit, ADOscore und ADOlog[11] (Lindemann et al., 2002) der BOC AG[12] einsetzen.

Die von der Plattform unterstützte Funktionalität, ihre intuitive Bedienung, Erweiterbarkeit und Stabilität hat sich sowohl in Industrie als auch Lehre und Forschung bewährt und war einer der Gründe warum sie für die Umsetzung der in dieser Arbeit entwickelten Methode herangezogen wurde. ADOxx ist speziell für den Entwurf und die Implementierung graphischer Modellierungssprachen entwickelt worden. Im Gegensatz beispielsweise zum Eclipse Modeling Framework (EMF) müssen für die Implementierung eines Metamodells in ADOxx keine Java Programmierkenntnisse vorhanden sein, denn die Konfiguration erfolgt hauptsächlich mittels eines graphischen Konfigurationstools (Fill et al., 2013a, S. 5). Des Weiteren bringt die Plattform notwendige Komponenten, wie ein Repository basierend auf einem Datenbanksystem, eine Benutzer- und Zugriffsverwaltung für Inhalt und Funktionalität, Client-/Server- beziehungsweise Cloud- Funktionalität und Algorithmen zur

[8] ADOxx ist eine registrierte Marke der BOC Products & Services AG.

[9] https://www.adoxx.org/live/home

[10] http://www.omilab.org

[11] ADONIS, ADOit, ADOscore und ADOlog sind registrierte Marken der BOC Products & Services AG

[12] http://www.boc-group.com

Auswertung und Dokumentation der Inhalte bereits mit. Andererseits ist ADOxx im Gegensatz zu EMF nicht Open Source sondern nur auf Open Use Basis einsetzbar. (Fill und Karagiannis, 2013). Sollen die Metamodelle durch zusätzliche Funktionalität, wie spezielle Auswertungs- und Ausführungsverfahren erweitert werden, so steht mit der Skriptsprache ADOscript (Junginger et al., 2000, S. 6) und (BOC Information Technologies Consulting AG, 2014) eine offene Programmierumgebung zur Verfügung die auch APIs zur Add-on Programmierung anbietet. Ein anderes, zu ADOxx ähnliches Tool ist MetaEdit+ das jedoch stark auf die automatische Generierung von Software Code fokussiert ist (Kelly et al., 2005).

Ein weiterer wesentlicher Grund für die Wahl von ADOxx für diese Arbeit ist, dass sich der Autor schon während seiner langjährigen Tätigkeit als Berater einen großen Erfahrungsschatz in der Umsetzung, Anpassung und Anwendung von Modellierungssprachen auf ADOxx aneignen konnte und somit eine effiziente prototypische Implementierung der Forschungsergebnisse ermöglicht wird.

Heutzutage bieten viele Modellierungstools die Funktionalität die Methode anzupassen und somit in bestimmter Weise eine Änderung des Metamodells vorzunehmen. Aber nur wenige Plattformen bieten so weitreichende Funktion Anpassungsmöglichkeit wie ADOxx.

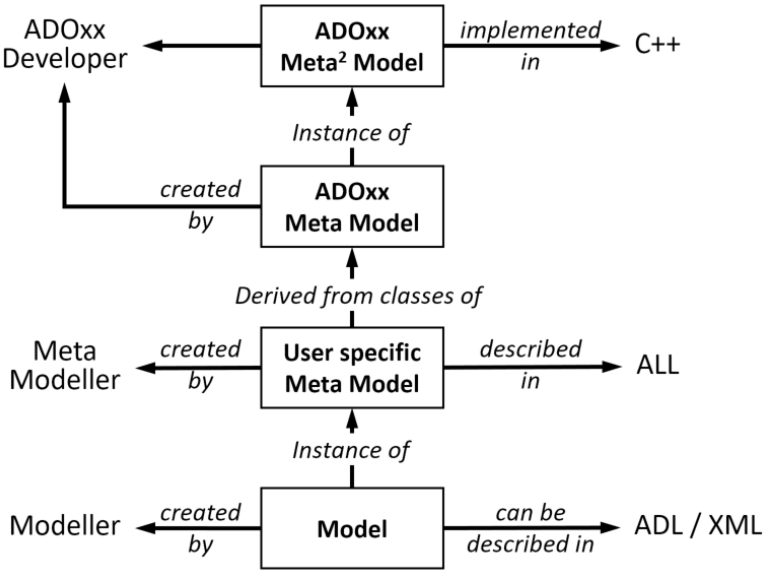


Abbildung 4: Nutzerrollen und Sprachen in der Metamodellierungshierarchie von ADOxx (Fill und Karagiannis, 2013)

Wie aus Abbildung 4ersichtlich, wird das ADOxx Meta²-Modell von ADOxx Entwicklern in C++ implementiert, es ist von ADOxx-Nutzern nicht veränderbar. Ein Klassenmodell des ADOxx Meta²-Modells ist in Kapitel Implementierung in Abbildung 60 abgebildet. Fill (Fill und Karagiannis, 2013) unterteilt diese Ebene in ein ADOxx Meta² Modell und ein daraus instanziiertes ADOxx Meta Model. Da diese Aufteilung aber für den Nutzer nicht transparent

ist, wird in dieser Arbeit nicht näher darauf eingegangen. Der Metamodellierer nutzt das graphische Konfigurationstool oder die Sprache ALL (ADOxx[13] Library Language) um ein benutzer- oder domänenspezifisches Metamodell zu erstellen und automatisch in den Modelleditor zu integrieren. Basierend auf diesem Metamodell erstellt der Modellersteller im graphischen Modelleditor Modelle, indem er Klassen und Relationen vom Metamodell in einem Modell als Objekte instanziiert. Der Modellnutzer betrachtet, analysiert oder verarbeitet die Modelle und kann sie beispielsweise im Fileformat ADONIS Definition Language (ADL) oder Extensible Markup Language (XML) exportieren.

2.2.2 Modellierungsmethoden-Engineering

Wie im Kapitel 2.1.2 Generische vs. Domänenspezifische Modellierungssprachen ausgeführt, muss vor dem Start eines Modellierungsvorhabens die zu verwendende Modellierungssprache festgelegt werden. Entscheidend für die Auswahl können unterschiedliche Kriterien sein, wie der Modellierungszweck, die Besonderheiten der Domäne, die Methodenaffinität der Benutzer oder falls das Modell in ein ausführendes Softwaresystem transformiert wird, dessen Anforderungen an Konfigurationsparameter. Die Analyse der Anforderungen kann ergeben, dass keine der bekannten Modellierungssprachen, ob nun generisch oder domänenspezifischer Natur, die speziellen Anforderungen des Modellierungsprojekts unterstützt und eine neue Sprache entworfen, beziehungsweise eine existierende Sprache um Konstrukte erweitert werden muss.

Neben der Sprache muss ein Vorgehen zur Modellierung festgelegt werden, um dem Projekt Mitarbeiter mit adäquaten Fähigkeiten, Wissen und Kompetenzen zuordnen zu können und bei Projekten mit mehreren Modellierern ein standardisiertes Vorgehen und somit gleichartige Ergebnisartefakte sicherzustellen. Wird eine neue Sprache entworfen, so muss vom Metamodellierer auch deren Verwendung beschrieben werden, um zu gewährleisten, dass die Modellierer die Sprache richtig anwendet und die Resultate den gewünschten Zweck erfüllen.

Des Weiteren muss das verwendete Werkzeug festgelegt werden, das durch Algorithmen und Mechanismen die Modellierer und Modellnutzer unterstützt. Soll eine neue Sprache oder Spracherweiterung in einer Software[14] genutzt werden, so muss dieses typischerweise konfiguriert werden.

---

[13] Ursprünglich „ADONIS Library Language“ genannt

[14] In dieser Arbeit wird davon ausgegangen, dass die Modellerstellung in einer Software erfolgt. Es gibt aber erfolgreiche Sprachen, wie zum Beispiel die „Value Stream Maps“ (Wertstromdesign), deren Modellierungsprozedur ausdrücklich die Verwendung von Papier und Bleistift empfiehlt (Rother und Shook, 2003, S. 4).

Nach Karagiannis und Kühn bilden die drei Komponenten, Modellierungssprache, Vorgehensmodell und Mechanismen und Algorithmen eine Modellierungsmethode. Ein Vorgehen zum Entwurf und Implementierung einer Modellierungsmethode wird in diesem Kapitel beschrieben. Passend dazu ist Brinkkempers (Brinkkemper, 1996) Definition für Methoden-Engineering: „Method engineering is the engineering discipline to design, construct and adapt methods, techniques and tools for the development of information systems.“ Er bezeichnet Methoden, die für ein bestimmtes Projekt oder eine Situation angepasst wurden als „Situational Methods“ (Henderson-Sellers und Ralyté, 2010).

Fill und Karagiannis (Fill und Karagiannis, 2013) bezeichnen die Umsetzung einer Modellierungssprache auf einer bestimmten Plattform als „Conceptualisation“. Die Qualität des Ergebnisses hängt dabei sehr stark von der Erfahrung des Metamodellierers und seinen Kenntnissen bezüglich der Einflussfaktoren, wie zum Beispiel der Fachlichkeit der Domäne, ihrer methodenspezifischen Anforderungen, den Fähigkeiten der zukünftigen Nutzer und den Eigenschaften der Zielplattform, ab. Aus dieser breitfächerten Vielzahl von Anforderungen die richtigen Entwicklungsentscheidungen zu treffen, ist für Fill und Karagiannis „more of an art than a science“. Nichts desto trotz kann diese Tätigkeit durch Anwendung eines strukturierten Vorgehensmodells unterstützt werden und somit die Chance auf eine erfolgreiche Konzeptualisierung der Modellierungssprache erhöht werden.

Im Folgenden wird der in dieser Arbeit verwendete Ansatz dargestellt.

2.2.3 Der OMiLAB Life Cycle

Der OMiLAB Life Cycle (Open Models Laboratory, 2014, S. 7) und (Götzinger et al., 2016, S. 64) ist ein Vorgehensmodell zur Konzeptualisierung von Modellierungsmethoden. Obwohl die Forschungstätigkeit zu der in der vorliegenden Arbeit vorgestellten Methode schon vor der Veröffentlichung des OMiLAB Life Cycles begann, ist das hier verwendete Vorgehen sehr ähnlich zum OMiLAB Life Cycle, da dieser auf Basis praktischer Erfahrungen und Erkenntnissen von Metamodellierern entstanden ist.

Das von OMiLAB vorgeschlagene Vorgehen besteht aus fünf Phasen, welche schrittweise durchlaufen werden und über Evaluierungsmechanismen zur Qualitätssicherung verfügen.

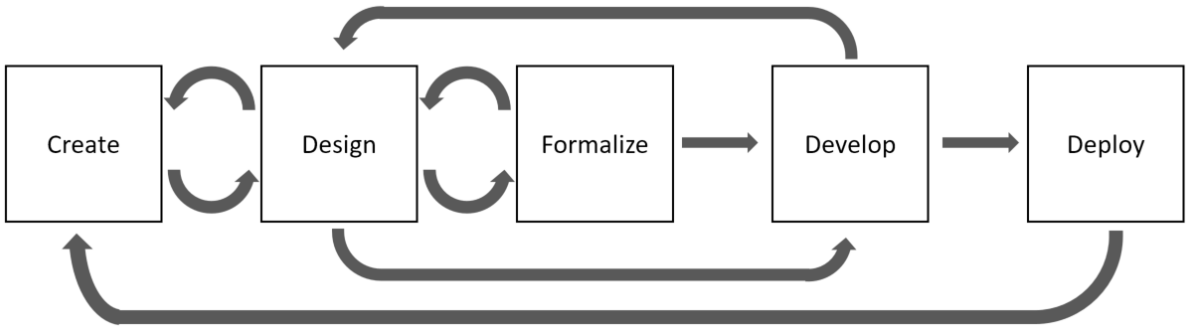


Abbildung 5: OMiLAB Life Cycle nach (Efendioglu et al., 2015)

2.2.3.1 Creation Phase

In dieser Phase wird die Domäne und Anwendungsszenarien der Modellierungsmethode abgegrenzt und der Modellierungszweck festgelegt. Der Metamodellierer erwirbt relevantes Wissen über den Bereich, wie domänenspezifische Konzepte und Begrifflichkeiten, indem beispielsweise:

- eine Literaturstudie durchgeführt wird,
- Spezialisten befragt werden,
- gegeben falls IT–Systeme analysiert werden und
- existierende Modellierungssprachen und Ontologien untersucht werden.

Daraus ergeben sich methodenspezifische Anforderungen bezüglich der Modellierungssprache, deren Anwendung und dafür notwendige Toolfunktionalitäten. Die Anforderungen werden beschrieben und können bei Bedarf priorisiert werden. Laut (Hinkelmann et al., 2015, S. 22) ist dieser Schritt vergleichbar mit den ersten Schritten eines Ontologie-Entwicklungsprojekts nach Noy und McGuinness (Noy und McGuinness, 2001).

Bezüglich der Anforderungserhebung und -bewertung unterscheidet Frank (Frank, 2013) zwei Phasen. Zunächst werden generische Anforderungen, beispielsweise bezüglich Simplizität, Verständlichkeit und Benutzerfreundlichkeit erhoben und anschließend spezifische Anforderungen identifiziert. Er schlägt dafür das in Abbildung 6 dargestellte Vorgehen vor.

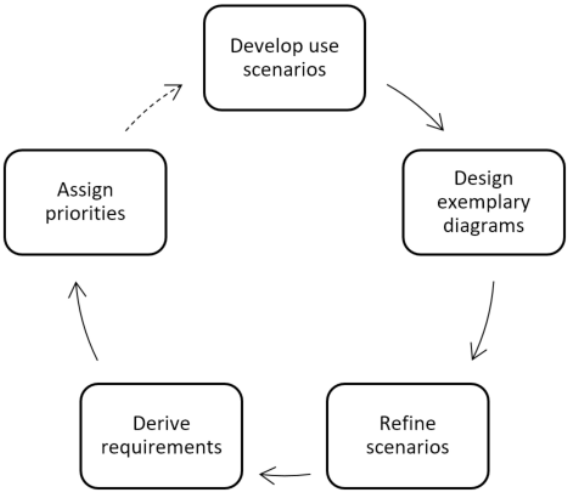


Abbildung 6: Micro Process "Analyse Spezieller Anforderungen" (Frank, 2013)

Basierend auf den Anforderungen wird ein konzeptionelles Modell der zukünftigen Modellierungsmethode erstellt, das zentrale Funktionalitäten, Schnittstellen und Standards beinhaltet.

2.2.3.2 Design Phase

In dieser Phase werden basierend auf den erhobenen Anforderungen und dem entwickelten Konzeptmodell die Bestandteile der Modellierungssprache, wie Notation, Syntax und Semantik entworfen und definiert. Ein weiteres Artefakt das in dieser Phase entsteht, ist ein plattformunabhängiges Metamodell. Zur Beschreibung dessen können beispielsweise UML Klassendiagramme verwendet werden. Götzinger et al. (Götzinger et al., 2016) fordert in dieser Phase auch eine Beschreibung des Metamodells in einer plattformunabhängigen Grammatik wie MM-DSL[15] (Visic et al., 2015).

Geforderte Funktionalitäten des Modellierungstools, wie zum Beispiel Mechanismen welche die Modellierung, Analyse oder Transformation der Modelle unterstützen, die nicht im Standard von der Plattform zur Verfügung gestellt werden, müssen hier detailliert spezifiziert werden.

Für die, in dieser Arbeit entwickelten Methode wurde ein agiler Ansatz gewählt und das Design entsprechend den Erfahrungen und Erkenntnissen aus der Arbeit mit dem, in der Development Phase (siehe Kapitel 2.2.3.4) entwickelten Prototypen verfeinert. Die formale Beschreibung des Metamodells erfolgte ebenfalls erst in der Formalization Phase und wurde in einer, an ADOxx angepassten Metamodellierungs-Beschreibungssprache, dem Formalism for Describing Meta-Models and Models (FDMM) (Fill et al., 2012) verfasst.

[15] MM-DSL (Modeling Method Domain - Specific Language) ist eine plattformspezifische Sprache zur Metamodelldefinition

2.2.3.3 Formalization Phase

Ziel dieser Phase ist Unklarheiten des Designs vor der Umsetzung zu erkennen und sicherzustellen, dass das plattformunabhängige Metamodell formal korrekt ist. Diese formale Prüfung kann gemäß (Hinkelmann et al., 2015, S. 22) mittels mathematischer Mitteln erfolgen, im dem beispielsweise FDMM (Fill et al., 2012) verwendet wird. Des Weiteren kann die formale Korrektheit auch durch Semantic Web Technologien wie RDF (Resource Description Framework) (W3C, 2014) oder mittels eines auf ADOxx erstellten Prototyps gezeigt werden.

2.2.3.4 Development Phase

In dieser Phase erfolgt die eigentliche Implementierung, indem das plattformunabhängige Metamodell in ein plattformspezifisches Metamodell übertragen wird. Am Beispiel von ADOxx bedeutet dies, dass mit der Konfiguration[16] des Metamodells innerhalb der Plattform auch automatisch ein Modelleditor und grundlegende Funktionalitäten eines Modellierungstools zur Verfügung stehen. Etwaige zusätzliche Funktionalitäten, die Mechanismen und Algorithmen, werden in dieser Phase implementiert.

Das Modellierungstool, inklusive Modellierungssprache und Mechanismen und Algorithmen wird von Nutzern getestet und die daraus gezogenen Erkenntnisse fließen als Feedback in die Design Phase (Kapitel 2.2.3.2) zurück.

2.2.3.5 Deployment / Evaluation Phase

In dieser Phase wird das Modellierungstool in ein verteilfähiges Softwarepaket integriert und einem größeren Nutzerkreis als Standalone-, Client-/Server oder einer Cloud-basierten Modeling-as-a-Service Lösung zur Verfügung gestellt. Ein Produktmanager oder nach Scrum-Rollenmodell ein Produkt Owner moderiert einen Feedbackprozess der Verbesserungen und Anforderungen für die nächste Version der Modellierungsmethode liefert.

2.2.4 Agile Model Method Engineering

Wie auch die Erfahrung des Autors, nicht zuletzt mit der Entwicklung der in dieser Arbeit vorgestellten Methode zeigt, ist die Konzeptualisierung einer Modellierungsmethode kein rein linearer, sequentieller Prozess, sondern vielmehr ein iterativer Prozess dessen Ergebnis durch schrittweises Erproben von Entwicklungsständen ständige an Qualität gewinnt. Denn auch wenn die Struktur dieser Arbeit ein geradliniges und zielorientiertes, sequenzielles Vorgehensmodell suggeriert, so wurden doch während der Entwicklung der Methode

[16] Diese Tätigkeit wird auch als „Customizing“ bezeichnet

verschiedene Konzepte erdacht, prototypisch in ADOxx implementiert, getestet, teilweise verworfen und das Gelernte in einem angepassten Konzept umgesetzt. Während der Arbeit mit dem Prototyp wurden neue Anforderungen identifiziert die in das Konzept miteinflossen.

Dieser Erkenntnis trägt auch Karagiannis (Karagiannis, 2015), (Buchmann und Karagiannis, 2015) Rechnung, indem er mit Agile Model Method Engineering (AMME) einen agilen Ansatz zum Methoden Engineering vorschlägt, der auf dem OMiLAB Life Cycle aufbaut.

Buchmann und Karagiannis (Buchmann und Karagiannis, 2015) führen des Weiteren aus, dass traditionelle Modellierungssprachen, die seit längerem von einem großen Nutzerkreis mit ähnlichen Abstrahierungsgrad verwendet werden, als relativ stabile Artefakte angesehen werden können. In neuen, oder sich stark verändernden, dynamischen Domänen hingegen, ergeben sich andauernd neue oder sich ändernde Anforderungen, denen eine Modellierungssprache gerecht werden muss. Diese Rahmenbedingungen verlangen nach einer agilen Entwicklungsmethode für Modellierungssprachen, welche die Entwicklungstätigkeit in einzelne, in sich abgeschlossen Pakete, sogenannte Sprints, aufteilt. Das Generic Modelling Method Specification Framework des AMME bietet dem Metamodellierer dabei einen ersten Ansatz zur Strukturierung und Planung der Arbeitspakete.

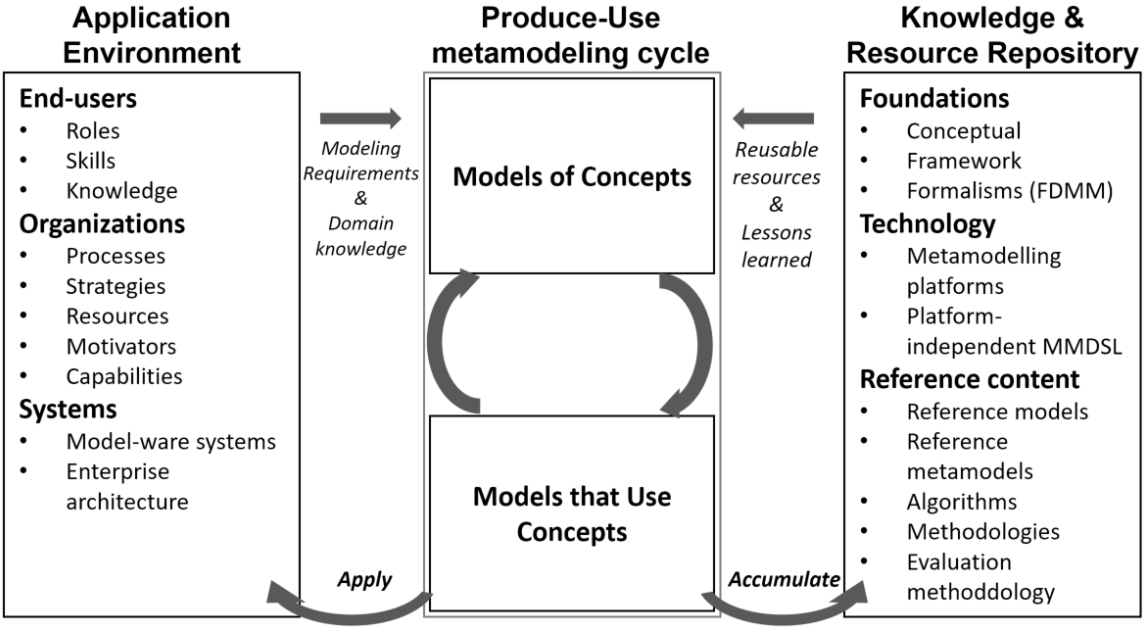


Abbildung 7: AMME Framework

Zentral für das in Abbildung 7 dargestellte AMME Framework ist der Produce-Use Metamodellierungs-Zyklus, der den agilen Charakter des Entwicklungsprozesses beschreibt. „Produce“ umfasst dabei den Entwurf, das Design und die prototypische Implementierung des Metamodells, und „Use“ umfasst das Testen und Evaluieren, also die Verwendung der Sprache, des Vorgehensmodells und der Mechanismen und Algorithmen zur Abbildung und Nutzung von Modellen.

2.3 Supply Chain Management

In der Literatur finden sich eine Vielzahl unterschiedlicher Definitionen der „Supply Chain“ und insbesondere des Konzepts „Supply Chain Management“. Eine Begründung hierfür ist, dass die zugrundeliegenden Verfahren zunächst in der Unternehmenspraxis entwickelt wurden und erst später, langsam von der Wissenschaft aufgegriffen wurden (Cooper et al., 1997). Einen guten Überblick verschiedener Definitionen und ihrer Herkunft bietet Mentzer et al. (Mentzer et al., 2001) und Stock et al. (Stock et al., 2010).

2.3.1 Begriffsdefinition

Trotz Auffassungsunterschiede verschiedener Autoren über den Begriff und Anwendungsbreite von Supply Chain Management haben Cooper et al. (Cooper et al., 1997) in einer Literaturstudie folgende Übereinstimmungen herausgearbeitet:

- Ein voll entwickeltes Supply Chain Management umfasst die gesamte Wertschöpfungskette vom Endkunden bis zum Lieferanten des Rohmaterials. Dafür ist eine starke Integration und Koordination der verschiedenen, in einer Supply Chain zusammenarbeitenden, Organisationen notwendig.
- Es umfasst mehrere, grundsätzlich voneinander unabhängig wirtschaftende, Unternehmen, die durch den Verbund in einer Supply Chain auf die Erlangung eines gemeinsamen Ziels hin arbeiten. Dennoch trachtet jedes Unternehmen danach seine eigenen Ziele zu erfüllen und kann Teil von mehreren, womöglich im Wettbewerb stehenden, Supply Chains sein. Die Pflege und Steuerung dieser organisationsübergreifenden Beziehungen und Schnittstellen ist von entscheidender Bedeutung.
- Supply Chain Management stellt die notwendigen Planungs-, Steuerungs- und Ausführungsaktivitäten bereit, um den bidirektionalen Fluss von Produkten, Dienstleistungen, Finanzmitteln und Information zu beherrschen.
- Es hat zum Ziel sowohl den Nutzen für den Endkunden, als auch den Erfolg der involvierten Unternehmen zu steigern.

Die dieser Arbeit zugrunde gelegte Definition der Supply Chain ist stark angelehnt an die Sichtweise von Christopher (Christopher, 2005, S. 17), Stadtler (Stadtler, 2005) und Mentzer et al. (Mentzer et al., 2001).

Eine Supply Chain ist ein Zusammenschluss von drei oder mehreren Organisationen, welche direkt in den wechselseitigen Material-, Informations- und Finanzfluss von der Quelle bis zum Kunden miteingebunden sind.

Organisationen können dabei eigenständige Unternehmen oder aber auch Teilbereiche eines Unternehmens sein. Werner (Werner, 2013) unterscheidet zwischen internen Supply Chains, welche innerhalb eines Unternehmens ablaufen, und netzwerkgerichteten Supply Chains, die verschiedene Unternehmen integrieren.

Der Materialfluss kann Rohmaterialien, Zwischenprodukte oder fertige Produkte enthalten.

Die Supply Chain ist der Betrachtungsgegenstand des Supply Chain Management, darunter ist zu verstehen.

Das Supply Chain Management umfasst alle Aktivitäten zur Integration von Organisationen entlang der Supply Chain und zur Koordination von Material-, Informations- und Finanzflüssen, um die Wettbewerbsfähigkeit der Supply Chain zu erhalten.

Da von einer Wettbewerbssituation ausgegangen wird, ist zur Erhaltung der Wettbewerbsfähigkeit eine ständige Verbesserung der Zielkriterien notwendig.

Der Begriff „Supply Chain“ suggeriert, dass die von der Wertschöpfungskette bedienten Märkte angebotsgetrieben sind, also einem Push-Prinzip unterliegen. Da aber viele Märkte nachfragegetrieben agieren, also dem Pull-Prinzip folgen, wäre hier eigentlich der Name „Demand Chain Management“ passender (Christopher, 2016). Des Weiteren handelt es sich oft nicht um eine geradlinige Kette (Chain) sondern eher um ein Netzwerk verschiedener Entitäten. Jede beteiligte Organisation oder jeder Organisationsbereich ist ein sogenannter Supply Chain Partner, dies inkludiert auch den Endkunden.

2.3.2 Supply Chain Partner

Eine erfolgreiche Supply Chain unterliegt dem Prinzip der Arbeitsteilung bei der jeder Teilnehmer, im Folgenden als Supply Chain Partner bezeichnet, seiner Spezialisierung entsprechend zur Wertschöpfung beiträgt. Partner können zum einen eigenständige Unternehmen oder zum anderen Bereiche eines Unternehmens sein. Eine Supply Chain muss bei einer sehr hohen Fertigungstiefe nicht unbedingt aus verschiedenen Unternehmen bestehen, sondern kann in einem Unternehmen angesiedelt sein. Die Partner sind dann die Tochterunternehmen, die beispielsweise an verschiedenen Standorten angesiedelt sind oder einzelne Unternehmensbereiche. Die Steuerung der Supply Chain wird damit einfacher, da anzunehmen ist, dass alle Partner dem übergeordneten Unternehmensziel folgen.

Lambert et al. (Lambert et al., 1998) unterscheiden dabei ausgehend von dem Unternehmen das im Fokus der Analyse steht, zwischen Kunden und Lieferanten, siehe Abbildung 8, die sie entsprechend dem Grad der Beziehung jeweils in „Tiers“ (Stufen) unterteilen. Des Weiteren sprechen sie von primären Partnern, die operationale oder steuernde Tätigkeiten in der Supply Chain durchführen, und unterstützende Partner, die Ressourcen bereitstellen, wie

zum Beispiel Energieversorger, Unternehmensberatungen und Verleihfirmen für Anlagen und Arbeitskräfte.

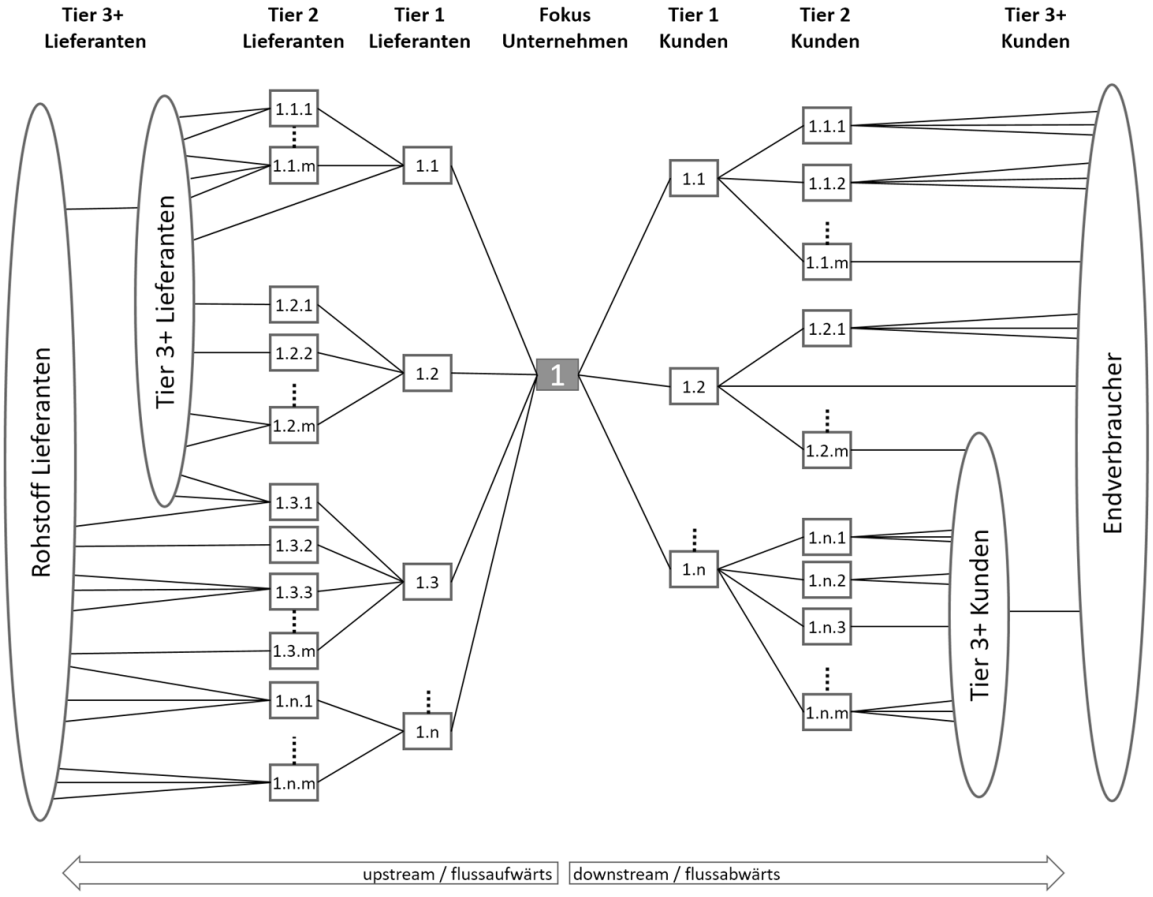


Abbildung 8: Supply Chain Netzwerk Struktur aus Sicht eines Unternehmens (Lambert et al., 1998)

Laut Stadtler (Stadtler, 2015, S. 10) gibt es zwei Hauptstoßrichtungen um die Wettbewerbsfähigkeit einer Supply Chain zu verbessern:

- Engere Integration oder Kooperation der involvierten Supply Chain Partner
- Bessere Koordination der Material-, Informations- und Finanzflüsse

2.3.3 Treiber des Supply Chain Managements

In Bezug auf Supply Chain Management sehen Kuhn und Hellingrath (Kuhn und Hellingrath, 2002) vier globale Treiber, die zu Veränderungen in Unternehmen führen:

1. Veränderung der Märkte durch Informations- und Kommunikationstechnologie (Geschwindigkeit)

Die horizontale Durchdringung der Märkte bewirkt, ermöglicht durch Kommunikationstechnologie, wie das Internet und mobile, soziale Anwendungen, eine grundlegende Veränderung im Käuferverhalten, sowohl im Endkundenbereich als auch im Firmenkundengeschäft. Die praktisch verzögerungsfreie Bestellmöglichkeit von Waren im Internet bringt die Erwartungshaltung mit sich, das auch die Lieferung schnellstmöglich erfolgt. Die Vergleichsmöglichkeiten von Preisen und Leistungen verschiedener Anbieter und deren Supply Chains erfordert die ständige Überprüfung und Verbesserung der Performance der Supply Chain. Dabei stellt die vertikale Durchdringung der Supply Chain mit Informationstechnologie eine Möglichkeit dar, dieser Anforderung nach Geschwindigkeit und Flexibilität zu begegnen. Sie erlaubt die bestehenden Geschäftsprozesse im Sinne der Zielkriterien zu verbessern oder gar neue Services dem Kunden anzubieten (Stadtler, 2005).

2. Wechsel von Push- zu Pull Märkten

In traditionellen Verkäufermärkten bestimmt der Hersteller welche Produkte, in welchen Mengen, zu welchem Preis zur Verfügung gestellt werden. Sie „pushen“ ihre Produkte in die Märkte und halten entsprechende Lagerbestände vor. Die zur Steuerung der Produktion und Logistik notwendigen Informationen wurden und werden hauptsächlich mithilfe von Planungs- und Prognoseinstrumenten ermittelt. Mit dem Wechsel zu einem Käufermarkt bestimmt der Kunde, welches Produkt er über welchen Verkaufskanal, in einigen Kanälen sogar zu welchem Preis, beziehen will. Dieser Trend zum „Pull“ Markt ist gepaart mit einem starken Anwachsen von Produktvarianten, individuellen Bestellungen und einer generell stärkeren Fokussierung auf den Kunden. Hierbei die Lagerkosten gering zu halten, dem Kunden aber dennoch in akzeptabler Zeit das Produkt bereitstellen zu können, ist eine Herausforderung an die Supply Chain und damit an deren Management. Aufgabe der Informationstechnologie ist es, so schnell wie möglich Informationen über das Käuferverhalten zu übermitteln, zu verarbeiten und entsprechende Aufträge an die betroffenen Supply Chain Partner zu versenden.

3. Steigende Komplexität

Supply Chain Netzwerke unterliegen dem Prinzip der Arbeitsteilung. Die verschiedenen Partner zeichnen sich beispielsweise durch Spezial-Knowhow, Kostenführerschaft oder Markenbekanntheit aus. Je mehr Partner im Netzwerk verbunden sind, desto höher wird die statische Komplexität (siehe Kapitel 2.3.4.3). Eine große Herausforderung stellt dabei die Planung und Steuerung der Supply Chain Partner, die gegeben falls weltweit verteilt sind, dar.

Um diese Komplexität im Ansatz beherrschbar zu machen und einen Überblick des Gesamtsystem zu erhalten, verweisen Warnecke und Braun (Warnecke und Braun, 2013) auf eine von Wahrnehmungspsychologen vorgeschlagene Vorgehensweise. Um in einem komplexen System übergeordnete Interdependenzen und Interaktionen zu erkennen und zu verstehen, sollte das System zunächst von einer gewissen Distanz, sie sprechen von „bewusst flüchtige Betrachtung“ beobachtet werden. Sobald diese Zusammenhänge erkannt sind, können einzelne Bereiche vertieft untersucht werden. Dies unterstützt die Forderung im Supply Chain Management, dass eine globale Optimierung des Netzwerks gegenüber einer lokalen Optimierung anzustreben ist.

4. Die Rolle der Dynamik

Neben der hohen statischen Komplexität der Supply Chain Netzwerke stellen die gestiegenen Anforderungen an Geschwindigkeit und damit eine schnellere Reaktionsfähigkeit auf Ereignisse, Kundenwünsche und Marktveränderungen (Stölzle und Karrer, 2003, S. 189) eine Herausforderung an Supply Chains dar. Die Fähigkeit zeitgerecht zu reagieren ergibt sich aus der Geschwindigkeit der Informationsverarbeitung in der Supply Chain, also wie schnell wesentliche Informationen erfasst, weitergeleitet, ausgewertet, korrekte Reaktionen ermittelt und diese vom adäquaten Supply Chain Partners umgesetzt werden können. Je kürzer die Planungshorizonte und je kleiner die Lagerbestände sind, desto sensibler reagiert die Supply Chain auf ungeplante Einflüsse, wie Störungen.

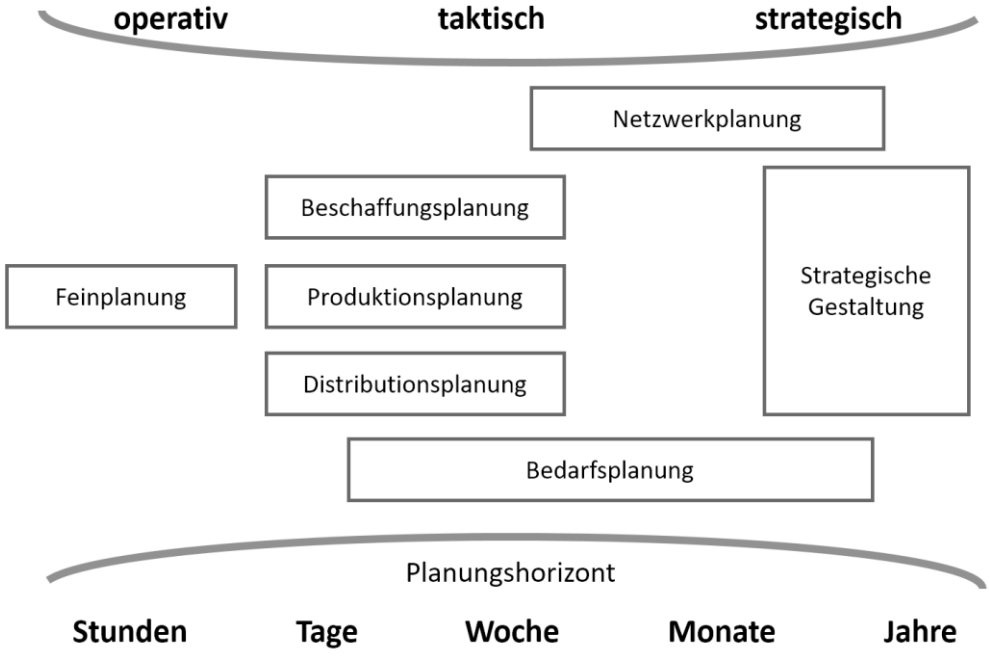


Abbildung 9: Zeithorizonte der Planung nach (Kuhn und Hellingrath, 2002)

2.3.4 Variabilität, Vernetztheit und Komplexität von Supply Chains

Für Robinson (Robinson, 2014) sind Variabilität, Vernetztheit und Komplexität eines Systems bestimmende Kriterien für den Aufwand der Modellbildung eines Realsystem und der Auswahl des geeigneten Analyseverfahrens. Bezogen auf die in dieser Arbeit maßgebliche Domäne der Instandhaltungs-Supply Chain bedeutet dies:

2.3.4.1 Variabilität

Offensichtlich wird die in einer Supply Chain auftretende Variabilität maßgeblich sowohl von internen als auch von externen Einflüssen bestimmt. Externe Einflüsse, können beispielsweise sich ändernde Kundennachfrage, Lieferengpässe von Lieferanten, Preisänderungen oder der wetterbedingte Ausfall von Transportrouten sein. Beispiele für interne Einflüsse sind Planänderungen, Stillstand von Maschinen wegen geplanter Wartung oder ungeplanten Ausfällen oder nicht synchronisierte Informations- und Materialflüsse. Sowohl interne als auch externe Einflüsse können vorhersehbar oder nicht vorhersehbar sein.

Da in dem entwickelten Ansatz nicht nur ein zeit-statisches Modell betrachtet wird, sondern die Ein- und Ausgabeparameter dynamisch im Zeitverlauf verfolgt werden müssen, spielt die Abbildung der Variabilität eine große Rolle. Robinson und Higton (Robinson und Higton, 1995) vergleichen anhand von einer Anlagenausfall-Szenarioanalyse die Behandlung der Variabilität in einem statischen und einem dynamischen Modell. Im statischen Fall müssen die Ausfallszeiten gemittelt den Prozessdurchlaufzeiten zugeschlagen werden, wohingegen im dynamischen Modell die Ausfälle direkt über stochastische Parameter modelliert werden. Sie zeigen, dass in diesem Fall das statische Modell falsche Ergebnisse produziert.

2.3.4.2 Vernetztheit

Supply Chains verbinden einzelne Supply Chain Partner. Ein hoher Grad an Vernetztheit liegt daher in der ‚Natur' dieser unternehmens- oder zumindest bereichsübergreifenden Prozesse.

Dabei sind die einzelnen Supply Chain Partner und deren Bearbeitungsstationen nicht nur über den Materialfluss, also die Weitergabe von Gütern, miteinander vernetzt, sondern in gleichem Masse über deren Informationsfluss. Darüber hinaus interagieren Material- und Informationsfluss miteinander an multiplen Stellen der Supply Chain und lösen damit in der Wertschöpfungskette Aktionen und Reaktionen aus. Zusammen mit den variablen Einflüssen macht ein höherer Grad der Vernetzung es schwieriger, das Verhalten eines Systems korrekt abzubilden und vorherzusagen.

Robinson (Robinson, 2014) stellt fest, dass der Fluss in operativen Systemen oft nicht nur in eine Richtung erfolgt, sondern oft über Schleifen und Feedback-Mechanismen verfügt, welche

die Zusammenhänge im Netzwerk sehr komplex machen können. Material und Güter fließen hauptsächlich in Richtung der Kunden, wohingegen die Information beispielweise über Lagerbestandsabfragen und Aufträge in die entgegengesetzte Richtung fließt.

2.3.4.3 Komplexität

Mit zunehmender Komplexität eines Systems steigt der Aufwand der Modellierung und damit die Gefahr der unzureichenden Abbildung des Systemverhaltens in einem Model. Es gibt dabei unterschiedliche Auffassungen ab wann ein System als komplex anzusehen ist. Wie Brooks und Tobias (Brooks und Tobias, 1996) in ihrer interessanten Diskussion zu diesem Thema anführen, reichen die Definitionen von, „komplex ist alles, womit wir Schwierigkeiten haben es zu verstehen“[17] (Flood und Carson, 2013) und (Golay et al., 1989) bis zu „einem System das aus einer großen Anzahl von Teilen besteht, die in einer nicht-einfachen Weise miteinander interagieren“[18] (Simon, 1991). Beide Definitionen können für Supply Chain Systeme zutreffen, speziell wenn sich die Interaktion der Teile im Zeitverlauf ändern kann. Diesen Aspekt beleuchtet Casti (Casti, 1979, S. 98ff) der zwischen einer statischen und dynamischen Komplexität unterscheidet und darauf hinweist, dass große Modelle nicht unbedingt eine hohe Komplexität aufweisen müssen, demgegenüber aber ein kleines Modell, mit einer geringen statischen Komplexität, eine hohe dynamische Komplexität haben kann. Dies gilt besonders dann, wenn zufällige Inputparameter das Modell beeinflussen. Sterman (Sterman, 1994) sieht eine Problematik in Modellen mit hoher dynamischer Komplexität, dass es zu Verzögerungen kommen kann, vom Zeitpunkt der Entscheidungsfindung bis zur Implementierung und damit über die Feedbackschleife erst verzögert Resultate zurückmeldet werden. Dies führt dazu, dass das System als Ganzes in seiner Lern- oder Verbesserungsfähigkeit eingeschränkt wird.

Um das zu illustrieren, entwickelte er mit der The System Dynamics Group am Massachusetts Institute of Technology's Sloan School of Management das Trainingsprogramm oder Spielsimulation „The Beer Distribution Game“ (Sterman, 1989)[19].

2.3.5 Flüsse innerhalb der Supply Chain

2.3.5.1 Der Materialfluss

Offensichtlich ist der Betrachtungsgegenstand des Materialflusses das Material das durch die Supply Chain bewegt wird. Als Material können zwar grundsätzlich alle Roh- und Hilfsstoffe,

[17] vom Autor aus dem Englischem übersetzt

[18] vom Autor aus dem Englischem übersetzt

[19] zur internetbasierten Durchführung vergleiche zum Beispiel Jacobs (Jacobs, 2000)

Betriebsmittel, Halbzeuge und daraus produzierte Teile verstanden werden, in dieser Arbeit wird das Material aber auf diskrete Objekte, welche wesentliche Bestandteile des Endprodukts sind, eingeschränkt. So werden beispielsweise Schmierstoffe, die zum Betrieb einer Maschine welche Teile bearbeitet, und Treibstoffe, die zum Betrieb von Fördermittels für diese Teile notwendig sind, nicht als Material betrachtet. Es wird davon ausgegangen, dass bei der taktischen oder operativen Betrachtung der Supply Chain diese Stoffe und Hilfsmittel als Teil der Bearbeitungsstationen oder Aktivitäten zur Verfügung gestellt werden.

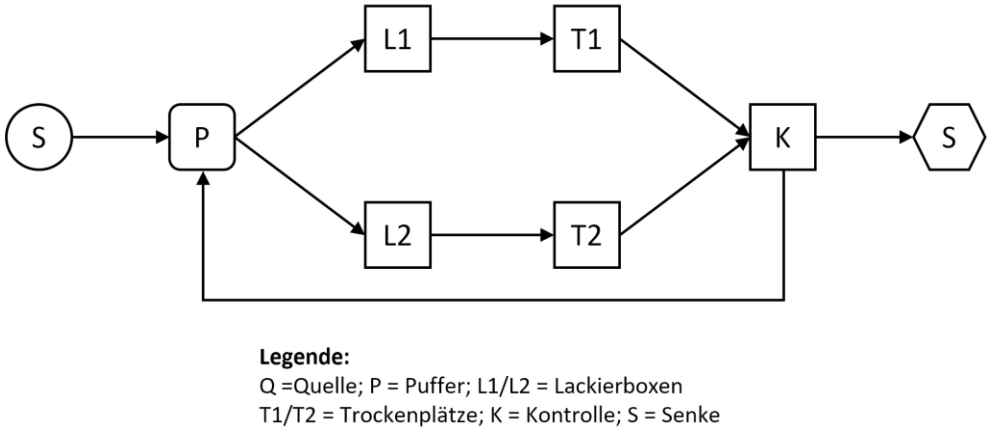


Abbildung 10: Beispiel für einen Materialfluss (Mebes, 2008)

In dieser Arbeit werden für diese im Materialfluss bewegten Teile synonym die Begriffe Gut oder Produkt verwendet. Teile können hierarchisch aufgebaut sein, beispielsweise wenn Teile als Komponenten zu einem Aggregat zusammengebaut werden oder zu Transporteinheiten verpackt werden müssen. Dementsprechend ändert sich im Materialfluss das zugrundeliegende Objekt. Am Beispiel eines Transportvorgangs erklärt, sind zu Beginn des Materialflusses eine Menge einzelner Produkte durch dieses Segment des Materialflusses zu bewegen, nach der Verpackung auf eine Palette reduziert sich die Betrachtung für das nächste Segments des Materialflusses auf die Palette und nach dem Entpacken müssen wieder die einzelnen Produkte bewegt werden.

Nach dem Verein Deutscher Ingenieure (VDI 2411) umfasst ein Materialfluss die Verkettung aller Vorgänge, in Bezug auf die Gewinnung, Be- und Verarbeitung sowie dem Verteilen von stofflichen Gütern, innerhalb eines festgelegten Bereichs.

Die angesprochenen Vorgänge können beispielsweise Fertigen, Entladen, Montieren, Sortieren, Fördern, Inspizieren, Verpacken oder Lagern sein. Allen gemein ist, dass sie zumindest eine Eigenschaft des im Materialfluss bewegten Gutes verändern (Günthner und Schneider, 2011).

Die Einschränkung auf stoffliche Güter ist dahingehend interessant, dass der Materialfluss somit nicht als Warenfluss für die „Ware“ Dienstleistung angesehen werden kann.

Üblicherweise werden dies diese Dienstleistungen als Geschäftsprozess zur Verfügung gestellt und nicht als Supply Chain.

Der in der Definition zitierte „festgelegte Bereich“ kann in der Domäne der Maintenance Supply Chain zu einem globalen, unternehmensübergreifenden Materialfluss führen.

Auch wenn die Bezeichnung „Fluss“ eine stetige Bewegung suggeriert, wird das Material in der Supply Chain häufiger pausieren, um beispielsweise auf den nächsten Bearbeitungsvorgang zu warten. Der Materialfluss ist eng an den unternehmensinternen Informations- und Datenfluss gekoppelt.

2.3.5.2 Der Informationsfluss

Informationsflüsse beeinflussen Material- und Finanzflüsse, sie planen, steuern und kontrollieren. Wie Material- und Finanzflüsse sind sie im Supply Chain Management überbetrieblich auszugestalten. Ein Literaturstudie zu Informationsflüssen in Supply Chains findet sich in (Madenas et al., 2014).

Informationen können separat von den Gütern (Heiserich et al., 2011, S. 337), die sie beeinflussen, fließen. Die Bewegung kann entsprechend ihres Informationsgehalts in die gleiche Richtung, wie zum Beispiel ein Lieferschein, oder in die entgegensetzte Richtung, wie zum Beispiel ein Auftrag, erfolgen. Oft stellt ein Gut an und für sich schon eine Information dar, so kann das Eintreffen eines Teils unverzüglich eine Bearbeitung auslösen, ohne dass zusätzlich ein Auftrag über das Informationsflusssystem bereitgestellt werden muss. Eine andere Möglichkeit der direkten Kopplung von Material- und Informationsfluss stellen Auto-ID Verfahren da, bei denen das Teil und der Datenträger miteinander fest verbunden sind.

Zumeist erfolgt die Erfassung, Verarbeitung und Übermittlung von Informationen EDV-unterstützt in Informationssystemen. Für unternehmensübergreifende Supply Chains müssen demgemäß einheitliche Datenübertragungsstandards vereinbart werden.

Die Beherrschung komplexer und dynamischer Supply Chain Netzwerke erfordert die sach- und zeitgerechte Information desjenigen, der Entscheidungen treffen und umsetzen muss. (Heiserich et al., 2011, S. 338). Oft müssen Ist-Größen mit Soll-Größen verglichen werden, wie zum Beispiel bei der Kontrolle des Lagerbestands, und gegeben falls Aufträge ausgelöst werden.

Der Informationsfluss einer Supply Chain kann in einzelne elementare Informationsflussbeziehungen unterteilt werden. In dieser Arbeit wird unter einer Informationsflussbeziehung eine unidirektionale Beziehung zwischen genau einem Sender und einem Empfänger verstanden. Über diese Beziehung können Informationen als Nachrichten vom Sender versendet werden, um den Empfänger beispielsweise vorbereitende

Informationen zur Verfügung zu stellen. Diese Art der Datenübermittlung wird auch als „push“ bezeichnet.

Eine andere Art der Informationsbeschaffung ist eine Abfrage, oder Abruf, eines Ist-Zustandes der je nach IT Möglichkeiten direkt als Service ausgebildet sein kann. Der Absender sendet dann nur eine Anfrage an die betreffende Stelle und erhält unverzüglich eine Antwort zurück, deshalb wird dies als „pull“-Prinzip bezeichnet.

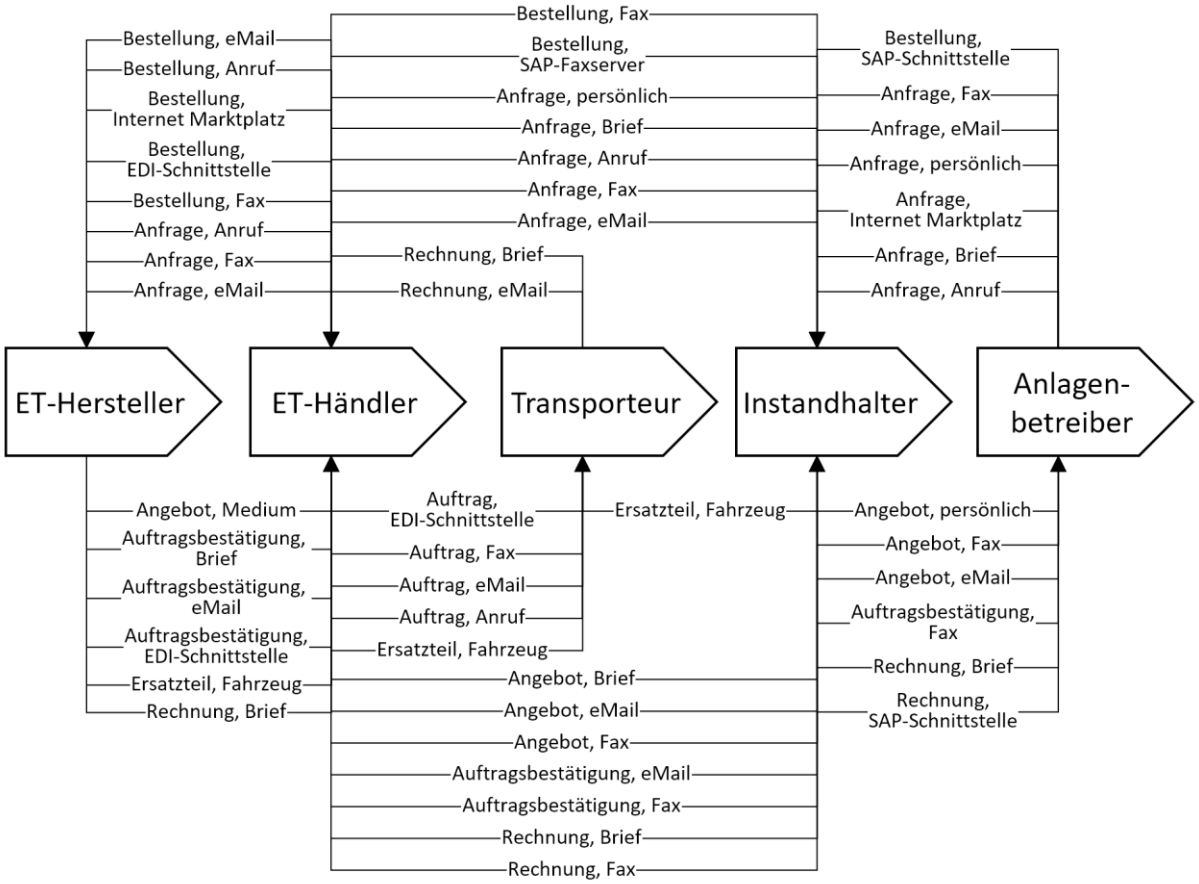


Abbildung 11: Beispiel für den Informationsfluss in einer Maintenance Supply Chain aus (Meier und Klimek, 2008)

2.3.6 Electronic Data Interchange (EDI) und Web-EDI

Der elektronische Datenaustausch, oder Electronic Data Interchange (EDI) ist der papierlose, automatische und elektronische Austausch von strukturierten Daten zwischen Anwendungssystemen von Geschäftspartnern (Heiserich et al., 2011, S. 356). EDI dient also dazu, die zwischenbetrieblichen Informationsflüsse zu integrieren. In der, in dieser Arbeit betrachteten Domäne sind das standardisierte technische und geschäftliche Daten, wie Abrufe, Arbeitsaufträge, Bestellungen, Lieferscheine, Bestandsabfragen die zwischen Supply Chain Partnern ausgetauscht werden (Werner, 2013, S. 283).

Ursprünglich handelte es sich um eine Punkt-zu-Punkt Kommunikation zwischen zwei Unternehmen, die oft auch proprietäre Austauschformate vereinbarten, welche den Informationsbedürfnissen ihrer Anwendungssysteme genügten (Steffen, 2001, S. 43). Da mit jedem weiteren Geschäftspartner ein Übertragungsprotokoll vereinbart werden musste und dies mit hohem Aufwand und Kosten verbunden war, wurden Standards zur Übertragung zwischenbetrieblicher Nachrichten entwickelt. Bekannte Standards sind beispielsweise ODETTE (Organization for Data Exchange by Teletransmission in Europe), das in der Europäischen Automobilindustrie verwendet wird und EDIFACT (Electronic Data Interchange for Administration, Commerce and Transport)

Wie aus Abbildung 12 ersichtlich, besteht ein EDI-System eines Unternehmens aus dem Konvertierungs- und dem Kommunikationssystem. Das Konvertierungssystem des Senders formt dabei das von den unternehmenseigenen Anwendungssystemen verwendete Format in das benutzte Standardformat um und sendet über das Kommunikationssystem die Nachricht an den Empfänger, wo diese, nach Erhalt, in dessen spezifisches Format konvertiert wird.

EDI erlaubt die schnellere und effizientere Übermittlung von Informationen und damit eine bessere Integration von Informationsflüssen verschiedener Supply Chain Partner. Da in einer Supply Chain mit vielen Partnern ein, wie bei EDI verfolgter, bidirektionale Aufbau des Kommunikationsnetzes jedoch hohe Kosten verursacht, wurde mit Web-EDI eine Lösung geschaffen, die eine n:m-Kommunikation über das Internet oder Extranet ermöglicht. Dabei bietet ein Partner Web-Seiten an, auf denen andere Partner der Supply Chain beispielsweise Abrufe, Bestellungen oder Rechnungen erstellen oder abrufen können (Heiserich et al., 2011, S. 360). Zur strukturierten Beschreibung der Geschäftsdaten wird dabei XML (eXtensible Markup Language) verwendet.

Diese im Vergleich zum klassischen EDI kostengünstigere Architektur erlaubt es auch mittelgroßen Unternehmen als Partner einer Supply Chain mitzuwirken und in den Informationsfluss miteingebunden zu werden.

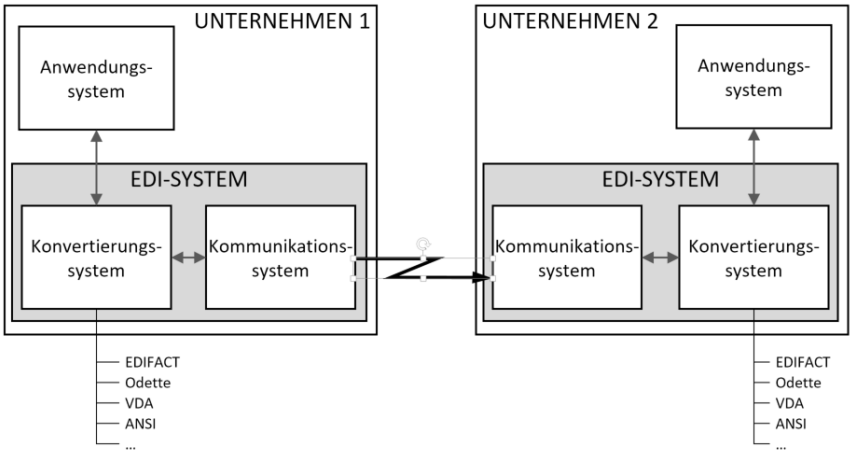


Abbildung 12: Grundstruktur eines EDI-Systems (Heiserich et al., 2011, S. 358)

2.3.7 Automatische Identifikations-Verfahren

Die eindeutige und schnelle Identifikation von Objekten ist eine Grundvoraussetzung für einen reibungslosen Materialfluss in der Supply Chain (Heiserich et al., 2011, S. 344). Um die Synchronisation von Material- und Informationsflüssen zu verbessern und sicherzustellen, dass relevante auftragsspezifische Informationen für ein individuelles Gut rechtzeitig vorliegen, kann das Stück mit dem Informationsträger direkt verbunden werden oder das Produkt an sich zum Informationsträger werden. Dies erfolgt seit Langem beispielsweise durch das Anbringen von Lieferscheinen auf der Produktverpackung oder das temporäre Notieren von Produktionsanweisungen direkt auf dem Stückgut.

Erfolgt die Identifikation von Objekten und die Erfassung und Weiterleitung von Daten an DV-Systeme vollautomatisch oder zumindest system-unterstützt, so spricht man von automatischen Identifikations-Verfahren. Diese werde auch „Automatic identification and data capture (AIDC)“ oder „Auto-ID“ genannt, bekannte Technologien hierbei sind unter anderem Optical Character Recognition (OCR), lineare oder 2-dimensionale Barcodes, Radio Frequency Identification (RFID), Magnetstreifen und Smart Cards und Biometrische Verfahren (Finkenzeller, 2015, S. 2).

Auto-ID kann verwendet werden um beispielsweise den Bestand an Waren im Lager zu verfolgen oder Produkte zu identifizieren. Auto-ID eröffnet auch die Möglichkeit, dass Objekte selbstständig Einfluss auf ihren Weg durch die Supply Chain nehmen. Ein Beispiel ist eine dringende Blutprobe im klinischen Labor, deren Tag ein Prioritätskennzeichen trägt, das dazu führt, dass sie vom System an Warteschlangen vorbei gelenkt wird. Das Objekt tritt sozusagen in Kontakt und interagiert mit den Systemen die es bearbeiten, beziehungsweise befördern. Wong et al. (Wong et al., 2002) gehen dabei soweit, dass sie von „intelligenten“ Produkten sprechen. Für Schuh (Schuh, 2006), (Schuh und Attig, 2009) ist dies Teil der „Sm@rt Logistics“.

Im Folgenden wird kurz auf die Vor- und Nachteile der Barcode-Technologie eingegangen und anschließend etwas ausführlicher die RFID Technologie erläutert, da diese beiden Verfahren im Besonderen für den Einsatz in Instandhaltungs-Supply Chain interessant sind.

2.3.7.1 Barcode

Neben der Optical Character Recognition (OCR) basiert die Barcode Technologie auf einer optischen, durch eine Maschine lesbaren Repräsentation von Daten. Traditionelle Barcodes oder Strichcodes sind eindimensional und codieren eine Identifikationsnummer. So besteht

der bekannte Standard zur Europäischen Artikel Nummer[20] (EAN) aus 13 Ziffern. Je nach Standard wird diese Identifikationsnummer oft erst durch Zusammenführen mit zusätzlichen, zentral gehaltenen Daten zur verwertbaren Information über ein Produkt. Um mehr Nutzinformation codieren zu können, wurden 2-dimensionale Codes, beispielsweise in Stapel-, Matrix- oder Kreisform entwickelt.[21] Diese zeichnet vor allem aus, dass die, gegenüber dem linearen Barcode höhere Informationsdichte es ermöglicht, Informationen zum Produkt direkt im Code zu integrieren. Eine Abfrage einer zentralen Datenquelle ist daher nicht immer notwendig. Des Weiteren steht mehr Speicherplatz für Prüf- und Korrekturzeichen zur Verfügung, was die Fehleranfälligkeit (beispielsweise durch Verschmutzung) des Scanprozesses reduziert. So kann beispielsweise der zur Verwendung auf Mobiltelefonen stark verbreitete QR-Code (Ohbuchi et al., 2004), ungefähr 4.000 alphanumerische Zeichen enthalten (Soon, 2008).

Der Inhalt des auf Produkten aufgebrachten Barcodes ist nicht veränderbar, Daten können weder gelöscht noch aktualisiert werden (sofern die Barcodes auf Papier oder das Produkt selbst gedruckt werden, es werden aber auch Barcodes auf Displays angezeigt, die dann natürlich veränderbar sind (Finkenzeller, 2015, S. 1). Ist es notwendig den Inhalt eines Barcodes zu ändern, so muss ein weiteres Etikett gedruckt und der bestehende Barcode überklebt werden. Für viele Einsatzszenarien, wie der Produktidentifikation oder Steuerung ist diese Einschränkung nicht bedeutend. Die kostengünstige Erstellmöglichkeit von Barcodes, Zuverlässigkeit und Ausgereiftheit des Verfahrens und die hohe Akzeptanz bei Kunden sind die entscheidenden Vorteile. Größte Nachteile sind, dass das Lesegerät eine direkte Sicht auf das Etikett benötigt und relative nahe an den Barcode geführt werden muss.

In der Domäne der Instandhaltung werden Barcodes verwendet, um die korrekten Ersatzteile oder Anlagen zu identifizieren. Da Barcodes an sich einfach kopiert werden können, bieten sie keinen Schutz vor Produktfälschungen, was bei kritischen Ersatzteilen problematisch sein kann (Lin et al., 2014).

Um mehr Daten direkt mit dem Produkt zu transportieren, ist die Verwendung von Speicherbausteinen in einem Siliziumchip die bessere Lösung. Typische Anwendungen dafür sind die Chipkarte mit elektrischen Kontakten, wie die Bankkarte oder die SIM-Karte für Mobiltelefone. Da die Kontakte der Karte zum Auslesen und Beschreiben die Kontakte des Lesegeräts berühren müssen, ist die Anwendungsbreite etwas eingeschränkt, diese Manko behebt die im Folgende beschriebene Radio Frequency Identification (RFID) Technologie.

---

[20] Auch als „International Article Number“ bezeichnet; Der UPC (Universal Product Code) ist eine Untermenge des EAN-Codes und somit kompatibel (Finkenzeller, 2015)

[21] Eine guten Überblick über Beispiele für verschiedene Barcodes bietet die Wikipedia Seite https://en.wikipedia.org/wiki/Barcode Abgerufen am 01.01.2017

2.3.7.2 Radio Frequency Identification (RFID)

Radio Frequency Identification (RFID) Technologie kam erstmals bereits im zweiten Weltkrieg zum Einsatz (Landt, 2005), (Werner, 2013, S. 286). Der mit ihrer Anwendung verbundene betriebswirtschaftliche Nutzen wurde aber erst in den späten 80er Jahren erkannt. Sie erlaubt die kontaktlose Identifizierung von Objekten und Erfassung oder den Austausch objektbezogener Daten ohne dass ein Sichtkontakt notwendig ist.

Ein RFID-System besteht aus drei Komponenten:

- RFID Transponder, auch kurz als „Tag“ bezeichnet, der am Objekt, dem Produkt direkt oder einem Behälter befestigt wird und die eigentlichen produktspezifischen Nutzdaten trägt. Ein Transponder besteht zumindest aus einem Mikrochip und einer Antenne.
- Lesegerät, oder ein Schreib-/ Lesegerät[22] und einem damit verbundenen
- Rechner, auf dem die Steuerungssoftware läuft

Zum Betrieb des Mikrochips im Transponder ist Energie notwendig, wird diese Energie aus den vom Lesegerät gesendeten elektromagnetischen Feldern gewonnen, so spricht man von passiven Tags. Verfügt der Tag über eine eigene Stromquelle ist es ein aktiver Tag (Werner, 2013, S. 287).

Die Near Field Communication (NFC) (Harnisch und Uitz, 2013) baut auf der RFID Technologie auf und eignet sich besonders für den kontaktlosen Datenaustausch über kurze Distanzen hinweg, sie findet besonders in mobilen Anwendungsszenarien wie Mobile Payment oder Ticketing Verwendung. Sie bietet eine höhere Flexibilität als RFID, da sie das aktive Lesegerät und den passiven Transponder in einem Gerät vereint und somit jedes NFC-taugliche Gerät sowohl aktive als auch passive Kommunikationsmodi unterstützt und somit die Kommunikation initialisieren kann. Die maximale Distanz zwischen zwei Geräten sollte unter 20cm liegen.

Auf die vielfältigen technischen Ausprägungen von RFID Transpondern und Lesegeräten wird hier nicht näher eingegangen. Finkenzeller bietet in seinem Standardwerk „RFID-Handbuch“ dazu einen ausgezeichneten Überblick (Finkenzeller, 2015).

Um die Integration der RFID Technologie mit den betrieblichen Geschäftsprozessen besser zu verstehend, wird im nächsten Kapitel ein Beispiel für eine RFID-Architektur beschrieben.

[22] In dieser Arbeit wird im Folgenden immer vom Lesegerät oder Reader gesprochen, auch wenn das Gerät in der Lage ist, Daten auf den Transponder zu schreiben

2.3.7.2.1 RFID-System Architektur

Bornhövd et al. (Bornhövd et al., 2004) sehen folgende Anforderungen an die Architektur von Auto-ID und somit RFID-Systeme

- Skalierungsfähigkeit bezüglich Anzahl von Tags und Lesegeräten
- Open System Architektur, die Verwendung von standardisierten Kommunikationsprotokollen wie TCP/IP, HTTP oder XML um größtmögliche Flexibilität bezüglich der verwendeten Hardware zu erhalten
- Effiziente Event Filter, um fehlerhafte oder redundante Lesevorgänge zu erkennen und zu unterdrücken und nur relevante Daten an die nachfolgenden Prozesse weiterzuleiten
- Eventbündelung, um das simultane Einlesen von mehreren Tags zu erlauben und diese effizient an die nachfolgenden Prozesse weiterzuleiten
- Flexibilität, um verschiedene Anwendungsszenarien abzudecken
- Verteilte Systemumgebung in verschiedenen Standorten, mobilen Endgeräten, lokal installiert oder Cloud-basiert
- Unterstützung für Systemadministration und Testing

Die in Abbildung 13 dargestellte beispielhafte RFID-System-Architektur besteht aus vier Schichten. Eine detaillierte Beschreibung des Modells findet sich in (Bornhövd et al., 2004). Der Device Layer bietet eine Hardware-unabhängige Schnittstelle um RFID-Module verschiedener Ausprägung verwenden zu können. Er stellt die elementaren Funktionen zur Verfügung, um Daten auf Tags zu schreiben oder auszulesen und diese an die nächste Schicht weiterzuleiten. Der Device Operation Layer koordiniert die verschiedenen Geräte, übernimmt die Event Filterung und Bündelung und bereitet die eingelesenen Daten für die nachfolgenden Prozesse auf. Der Business Process Bridging Layer ordnet die eintreffenden Ereignisse den bestehenden Geschäftsprozessen zu, pflegt die Statusinformation der im System mittels RFID verfolgten Objekte und bereitet die eingelesenen Daten und Events mittels Geschäftsregeln (Business Rules) für die Weiterverarbeitung in den Enterprise Anwendungen vor und leitet diese an die Anwendungen des Enterprise Application Layers weiter. Ein weiteres Beispiel für eine konzeptionelle RFID und EPCglobal (Electronic Product Code) Netzwerk Architektur findet sich in (Jakkhupan et al., 2011, S. 950).

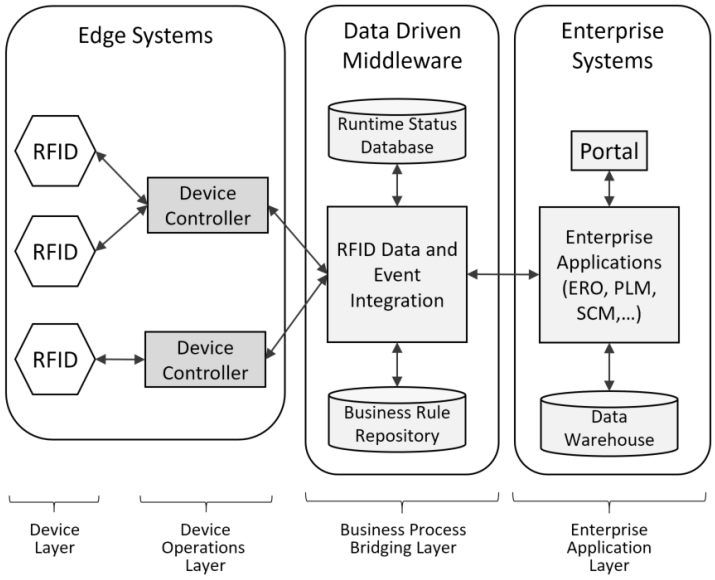


Abbildung 13: Beispiel einer RFID-Architektur; angepasst aus (Bornhövd et al., 2004) und (Zhao et al., 2012)

2.3.7.2.2 Kennzeichnungs- und Identifikations-Ebenen

Je nach Anwendungsgebiet werden RFID-Tags zur Kennzeichnung und Identifikation von Objekten auf verschiedenen Ebenen eingesetzt (Finkenzeller, 2015, S. 288):

Unit-Level: Werden die Produkte auf einem Ladungsträger, wie zum Beispiel einer Palette, transportiert oder gelagert, so kann der Ladungsträger mit einem Tag versehen werden. Sind in oder auf einem Träger nur Produkte desselben Typs und gegeben falls anderen deckungsgleichen Eigenschaften, wie Produktionscharge oder Haltbarkeitsdatum, gepackt, so können diese Produktdaten auf dem Transponder gespeichert werden. Befinden sich in oder auf einem Ladungsträger Produkte verschiedenen Typs, so wird am Tag üblicherweise nur die Unit-ID gespeichert und die Details über die enthaltenen Produkte werden in einer zentralen Datenbank vorgehalten. Für wiederverwendbare Ladungsträger kommen wiederbeschreibbare Tags zum Einsatz.

Case-Level: Ein Case kann ein Karton oder ein wiederverwendbarer Behälter sein und die Kennzeichnung mit RFID-Tags ist ähnlich der auf Unit-Level, mit dem Unterschied, dass der Transponder hier auf dem Karton oder Behälter platziert wird.

Item-Level: Ist der Tag direkt am Produkt befestigt oder integriert, spricht man von einer Kennzeichnung auf Item-Level. Typischerweise werden dafür Read-Only oder Write-Once-Read-Many Transponder verwendet, welche eine eindeutige Artikelnummer enthalten, die nach dem Auslesen mit einer Datenbank abgeglichen werden muss, um die relevanten Produktdaten zu erhalten.

2.3.7.2.3 Vor- und Nachteile der RFID Technologie

In vielen Publikationen werden qualitative Studien herangezogen, um den wirtschaftlichen Nutzen von RFID zu zeigen. (Leung et al., 2007, S. 52), die Vor- und Nachteile des Einsatzes von RFID werden im Folgenden benannt.

Vorteile der RFID Technologie gegenüber von Barcodes sind:

- RFID Tags können von einer größeren Distanz ausgelesen werden
- Sie können große Datenmengen enthalten, beispielsweise Wartungsinformationen, wie Historie oder Anleitungen
- Sie können mit hoher Geschwindigkeit ausgelesen werden, auch mehrere Tags zur gleichen Zeit
- Sie sind wiederbeschreibbar (es gibt auch nur einmal beschreibbare Tags)
- Inhalte können verschlüsselt werden
- Keine freie Sicht auf den Tag notwendig
- Nicht sehr empfindlich gegenüber Umwelteinflüssen

Nachteile der RFID Technologie gegenüber von Barcodes sind:

- Auch wenn Kosten sinken, sind Tags immer noch teurer als Barcode Etiketten[23]
- Abschirmung durch Metalle und Flüssigkeiten kann Schwierigkeiten beim Auslesen/ Beschreiben bereiten
- Das gleichzeitige Auslesen von mehreren Tags erfordert eine Kollisionsstrategie

Der Vergleich von Barcode und RFID zeigt, dass beide Technologien Vorteile haben, die sie für verschiedene Einsatzszenarien zur bevorzugten Alternative machen. Abbildung 14 aus Jakkhupan et al. (Jakkhupan et al., 2011) enthält die Darstellung eines Materialflusses und links und rechts die notwendigen Bearbeitungsschritte für Barcodes und die für RFID. Auch wenn die RFID-basierten Tätigkeitsschritte etwas idealisiert dargestellt werden, so ist doch der höhere mögliche Automatisierungsgrad dieser Technologie gegenüber Barcodes, die viele manuellen Schritte zum Scannen der Etiketten benötigt, ersichtlich.

[23] Asif (Asif, 2005) diskutiert einige Kostenfaktoren in seinem Paper

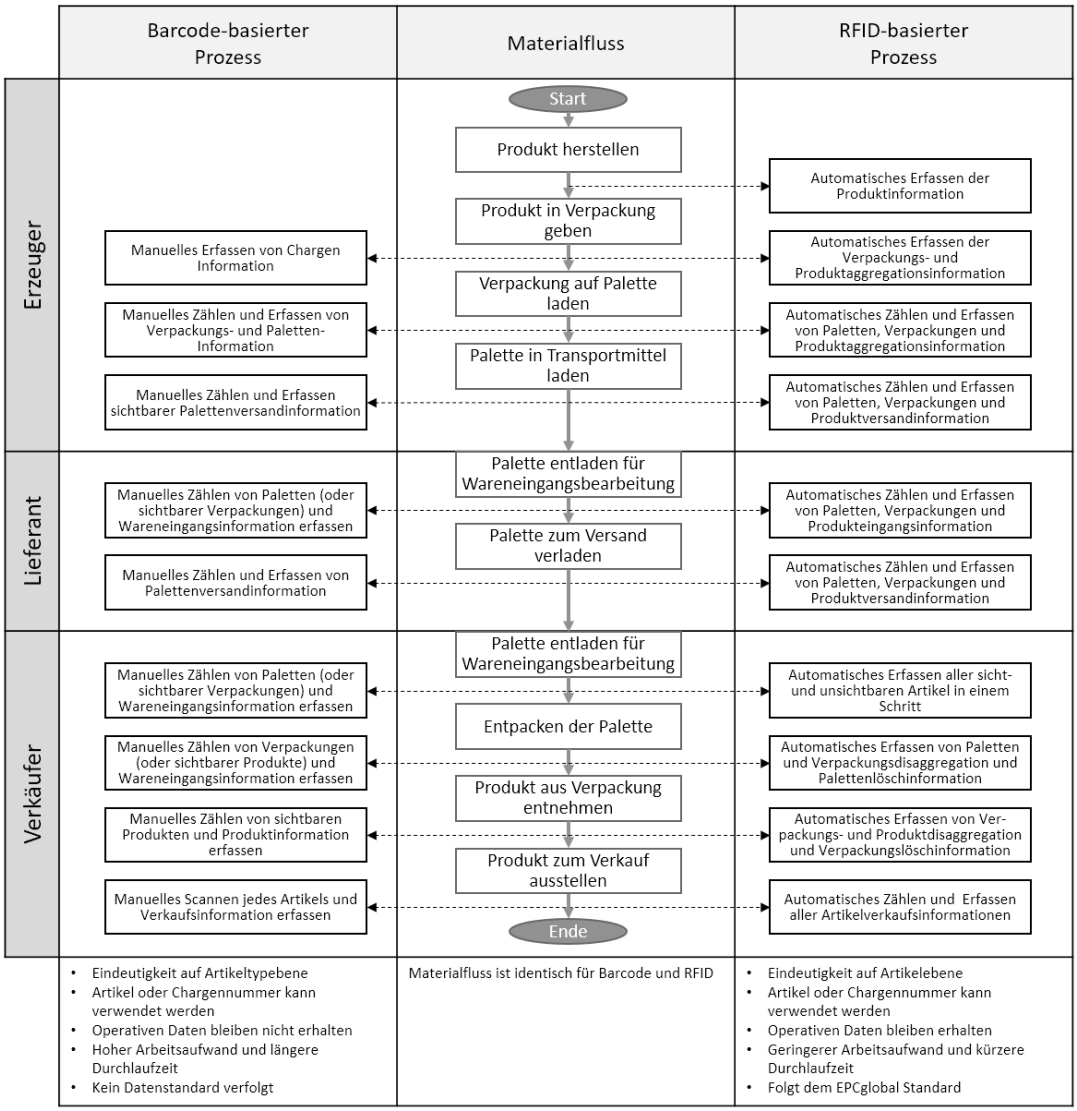


Abbildung 14: Geschäftsprozessanalyse zweier Auto-ID Verfahren (Jakkhupan et al., 2011)

2.3.8 SCOR

Eine typische Zielsetzung von domänenspezifischen Referenzmodellen ist die Definition eines Standards bezüglich Struktur und Terminologie der Fachlichkeit (Stölzle, 2005). Werner (Werner, 2013, S. 65) sieht dadurch die Entstehung einer „normierten Sprache für interne und netzgerichtete Kommunikationsprozesse“ gefördert.

2.3.8.1 Grundlagen zum SCOR Model

Im Supply Chain Management hat sich dabei das SCOR (Supply Chain Operations Reference Model) durchgesetzt. Weitere Referenzmodelle sind beispielsweise GSCF (Global Supply Chain Framework) (Lambert et al., 1998), ein von Mentzer et al. (Mentzer et al., 2001) vorgeschlagenes und LogiBest (Bruckner und Spille, 2004), ein Referenzmodell logistischer

Prozesse, das ein branchenübergreifendes Benchmarking erlaubt. Einen guten Überblick über verschiedene Supply Chain Referenzmodelle bieten (Ponis et al., 2013) und (Naslund und Williamson, 2010).

Das SCOR Model wurde ursprünglich vom Supply Chain Council (SCC) entwickelt, einem branchenübergreifenden Verband von Unternehmen und Organisationen welcher 1996 gegründet wurde. Seither wurde es stetig weiterentwickelt und den Veränderungen und Anforderungen des Supply Chain Managements angepasst, aktuell liegt Version 11 des SCOR Modells vor. In 2014 ging die APICS SCC[24] aus einem Zusammenschluss der APIC (American Production and Inventory Control Society) und dem SCC (Supply Chain Council) hervor, die seither die Weiterentwicklung und globale Vermarktung übernimmt.

Das SCOR-Modell erstreckt sich über die gesamte Supply Chain von der initialen Quelle bis zum Endverbraucher. Da es branchenübergreifend konstruiert wurde, können verschiedene Supply Chain Konfigurationen von dem Modell abgedeckt werden. Darüber hinaus enthält es nicht nur logistische Prozesse sondern auch Planungs-, Steuerungs- und Produktionsprozesse. Durch die breite Abdeckung der Thematik ist es für diese Arbeit von besonderer Bedeutung.

2.3.8.2 Struktur des SCOR Modells

Das SCOR Referenzmodell besteht aus vier Komponenten:

Performance: Stellt standardisierte Kennzahlen zur Verfügung, um die Prozessleistung zu beschreiben und um strategische Zielsetzungen der Supply Chain zu definieren

Prozesse: Beschreibt standardisierte Prozesse und deren Interaktionen

Best Practice: Beschreibt bewährte Verfahren, die eine signifikant bessere Leistung der Supply Chain bewirken

Personal: Definiert benötigte Fähigkeiten und Kompetenzen, um die beschriebenen Supply Chain Prozesse auszuführen

Da sich diese Arbeit hauptsächlich mit der Simulation von Prozessen befasst wird im Folgenden nur die Prozess-Komponente beschrieben.

[24] siehe Webpage http://www.apics.org/about/overview/about-apics-scc am 23/12/2016

	Ebene		Beispiele	Bemerkung
	Nummer	Bezeichnung		
In SCOR enthalten		Prozesstypen	• Plan • Source • Make • Deliver • Return und • Enable	In Ebene 1 werden der Umfang und Inhalt der Supply Chain definiert und Leistungsziele definiert
		Prozess-kategorien (Konfiguration)	• Make-to-Stock • Make-to-Order • Engineer-to-Order	Ebene 2 definiert die operativen Strategien und legt die dafür notwendigen Prozesse fest
		Prozessschritte	• Plane Lieferung • Nehme Ware an • Prüfe Produktqualität • Autorisiere Zahlung	Ebene 3 bestimmt die Fähigkeit die Prozesse erfolgreich auszuführen und beschreibt: • Prozesse • Inputs und Outputs • Prozessperformance • Technische Fähigkeiten • Fertigkeiten des Personals
Nicht in SCOR enthalten		Aktivitäten (Implemen-tierung)	Branchen-, Unternehmens-, Lokations- und/oder Technologie-spezifische Aktivitäten	Ebene 4 beschreibt die Aktivitäten die in einer Supply Chan durchgeführt werden im Detail. Unternehmen implementieren spezifische Prozesse und Verfahren um die geforderten Outputs und Performanz zu erreichen

Abbildung 15: Prozessebenen des SCOR Modells; verändert übernommen aus (APICS Supply Chain Council, 2015)

Das SCOR Modell ist in vier hierarchische Ebenen strukturiert deren Detaillierungsgrad mit fortschreitender Tiefe steigt.

2.3.8.2.1 Ebene 1 – Prozesstypen

Diese Ebene führt sechs unterschiedliche Prozesstypen ein, welche die gesamte Lieferkette abbilden. Aus Sicht eines Supply Chain Partners gibt es an organisationsübergreifenden Schnittstellen materialflussaufwärts zumindest einen Lieferanten und abwärts zumindest einen Kunden (siehe Abbildung 16). Ein Prozesstyp kann dabei als eine Gruppe oder ein Bündel von Prozessen mit gleichartiger Ausrichtung verstanden werden, diese sind (Supply Chain Council, 2010), (APICS Supply Chain Council, 2015):

Plan (Planen): Generische Planung und Steuerung der Supply Chain. Erstellt Pläne, um die Menge, Qualität und Preis der von der Supply Chain angebotenen Waren der voraussichtlichen Nachfrage anzupassen und reagiert auf plötzliche Schwankungen und Störungen. Wie aus Abbildung 16 ersichtlich, kann die Planung zentral beziehungsweise integriert (Hofmann et al., 2012, S. 238) und / oder lokal erfolgen.

Source (Beschaffen): Prozesse, die zur Deckung von geplanten und aktuellen Bedarfen dienen.

Make (Produzieren): Prozesse zur Herstellung oder Transformation von Gütern und Zwischenprodukten die durch die Supply Chain bewegt werden und den Materialfluss bilden. Als generelle Richtlinie wird im SCOR Referenzmodell genannt, dass in einen Make-Prozess ein oder mehrere Teile als Input fließen können, welche ein oder mehrere Teile als Output ergeben. Dies gilt offensichtlich nicht immer für Make-Prozesse die Dienstleistungen, wie zum Beispiel eine Inspektion, abbilden.

Deliver (Liefern): Inkludiert alle Aktivitäten die mit der Auslieferung von Erzeugnissen zum Kunden in direktem Zusammenhang stehen.

Return (Rückführen): Beinhaltet alle Tätigkeiten im Bereich der Rückgabe von Waren, Behältern oder Rohstoffen von Kunden an Lieferanten.

Enable (Bereitstellen): Prozesse dieses Typs sorgen dafür, dass Geschäftsregeln definiert sind, die Supply Chain Performance überwacht wird, die benötige Informations- und Datenverarbeitung Infrastruktur zur Verfügung steht, die Daten gepflegt werden und passende personellen Ressourcen bereitgestellt werden.

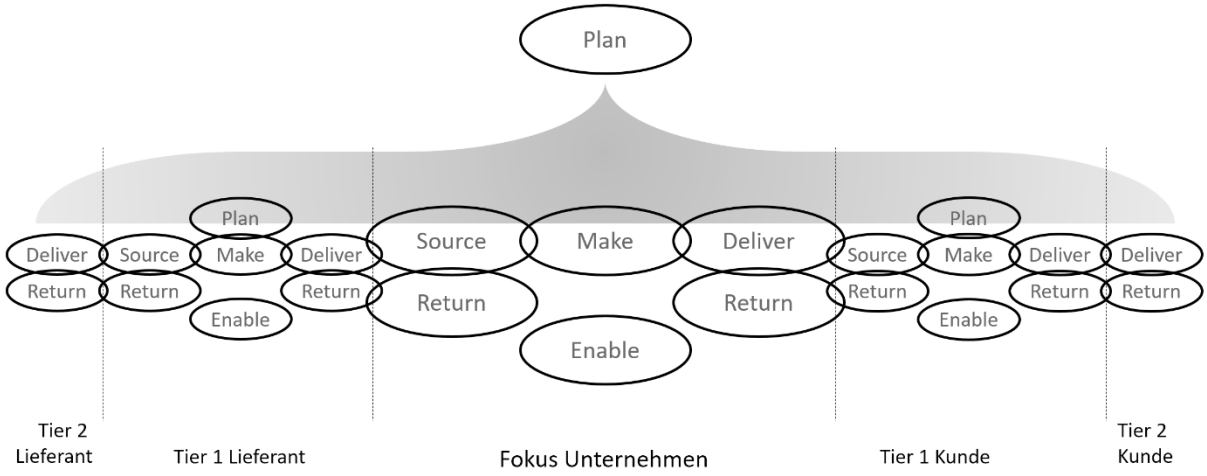


Abbildung 16: SCOR-Modell, (Supply Chain Council, 2010)

2.3.8.2.2 Ebene 2 – Konfigurations-Ebene

Die Konfigurations-Ebene bestimmt die Leistungsfähigkeit der Level 1 Prozesse. Die wichtigsten Level 2 Prozesse für die Prozess-Typen Source, Make und Deliver sind Lagerfertigung oder MTS (Make-to-Stock), Auftragsfertigung oder MTO (Make-to-Order) und kundenspezifische Auftragsfertigung oder ETO (Engineer-to-Order) (Heiserich et al., 2011, S. 18).

Auf dieser Ebene werden für jeden Prozess Kennzahlen vorgeschlagen und Best Practices empfohlen. So wird beispielsweise für den Plan Prozess ein ereignisgesteuertes Supply Chain Re-Planning vorgeschlagen, das zu einer unverzüglichen Änderung der Produktions-und Beschaffungspläne führt, sobald eine Veränderung der Nachfrage identifiziert wird (Supply Chain Council, 2010, S. 3.1.2). Für die Lagerfertigungsvariante des Make-Prozesses werden Verfahren, wie VMI (Vendor Managed Inventory) oder Demand-Pull Manufacturing, beispielsweise mittels Kanban beschrieben.

Für die Return-Prozesse werden drei Ausprägungen definiert. Je eine für defekte Produkte, überschüssige Produkte und zu wartende oder zu reparierende Produkte (wird im Modell als „MRO (maintenance, repair, and overhaul)“ bezeichnet).

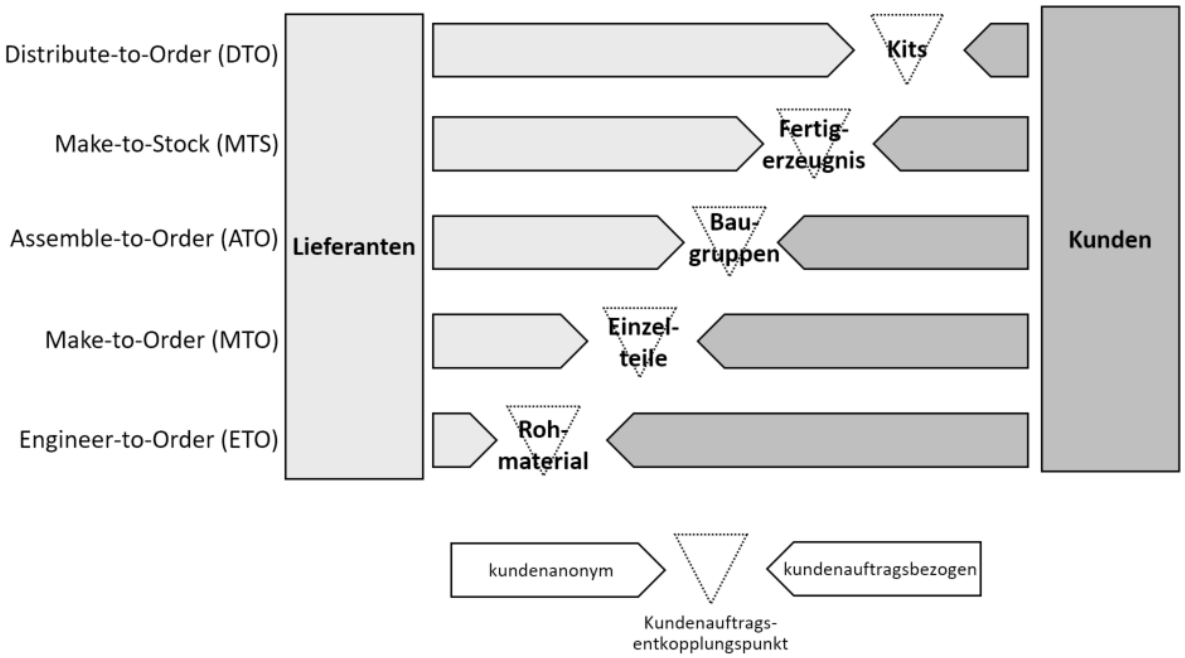


Abbildung 17: Produktionstypologien (Heiserich et al., 2011, S. 37)

2.3.8.2.3 Ebene 3 – Prozessschritt-Ebene

Ebene 3 detailliert die auf Ebene 2 beschriebenen Prozesse in einzelne sequentielle Tätigkeitsschritte und ihre Input- und Output Beziehungen zu anderen Prozessen des Modells. Angelehnt an ein Beispiel von Werner (Werner, 2013, S. 69) und dem SCOR Model

Version 10.0 (Supply Chain Council, 2010) wird ein Ausschnitt des Prozesses der Kundenspezifischen Auftragsfertigung diskutiert:

- Produktionsaktivität terminieren (Schedule Production Activities sM3.2)
- Material ausgeben (Issue Sourced/ In-Process Product sM3.3)
- Produktion und Test (Produce and Test sM3.4)
- Verpacken (Package sM3.5)
- Produkt bereitstellen (Stage Finished Product sM3.6)

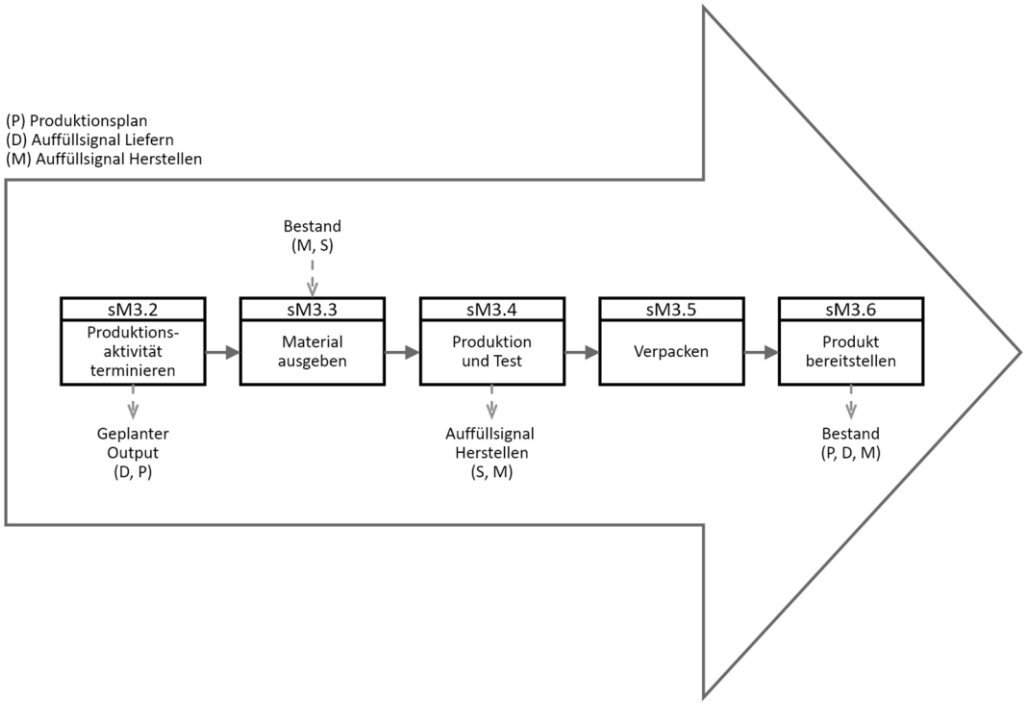


Abbildung 18: SCOR Prozess zur Kundenspezifischen Auftragsfertigung

Treiber dieser Kausalkette ist der Prozessschritt zur Terminierung der Produktionsaktivität sM3.2. Als Input für diesen Schritt muss das Informationsflusssystem unter anderem den Produktionsplan (erstellt im Plan-Prozess), Auffüllsignale zur Lieferung (Deliver-Prozess) oder Lagerhaltung und Aufträge zur Herstellung (Make-Prozess) bereitstellen. Nach Verarbeitung dieser Eingangsfaktoren kann eine Information zum erwarteten Produktionsoutput an den Plan- und Deliver-Prozess gegeben werden.

Das Bestandssignal, das aus einem Source oder Make-Prozessschritt generiert wird, triggert nachfolgend die Ausgabe (sM3.3) von dem für die Produktion benötigten Material. Mit der Bereitstellung des Materials über den Materialfluss startet der Produktions- und Test-Schritt (sM3.4). Nach Abschluss der Produktion stellt das System eine Bestandslücke fest und sendet ein Signal „Auffüllsignal Herstellen“ an den Source- und / oder Make-Prozess über das Informationsflusssystem. Im nächsten Schritt wird das produzierte Gut verpackt, bereitgestellt und entsprechend der Position dieses Prozesses in der Supply Chain der Deliver-, Plan- oder nachfolgende Make-Prozess über den Bestand informiert.

2.3.8.2.4 Ebene 4 Implementierungs-Ebene

Diese Ebene stellt die Prozessschritte in höherer Konkretisierung dar. Da bei diesem Detailgrad unternehmens- und branchenspezifische Besonderheiten die Prozesse prägen, verzichtet das SCOR Model auf eine Definition der Inhalte dieser Ebene. Sollten weitere Detaillierungsebenen erforderlich sein, stellt es das SCOR Model frei eine fünfte oder sechste Ebene zu verwenden.

2.3.9 Beispiele für Supply Chain Konfigurationen

Das SCOR Referenzmodell enthält sowohl Vorgaben zur Struktur des Prozessmodells als auch inhaltliche Definitionen von Supply Chain Prozessen und Kennzahlen. Umeda und Zhang (Umeda und Zhang, 2006) und (Umeda und Zhang, 2008, S. 454) präsentieren ein strukturelles Modell mit dem Namen "Feature-elements model" dessen Konzepte in Kapitel 2.3.9.1.1f näher beschrieben werden. Li und Wong (Li und Wang, 2007) bieten in ihrem Aufsatz einen Literaturüberblick über Supply Chain Konfigurationsmechanismen. Im Folgenden werden die von Umeda und Jain (Umeda und Jain, 2004) aufgezeigten alternativen Supply Chain Konfigurationen erläutert. Das Modell ist deshalb von besonderem Interesse für diese Arbeit, da sie es als Grundlage für eine ereignisgesteuerte Simulation entwickelt haben.

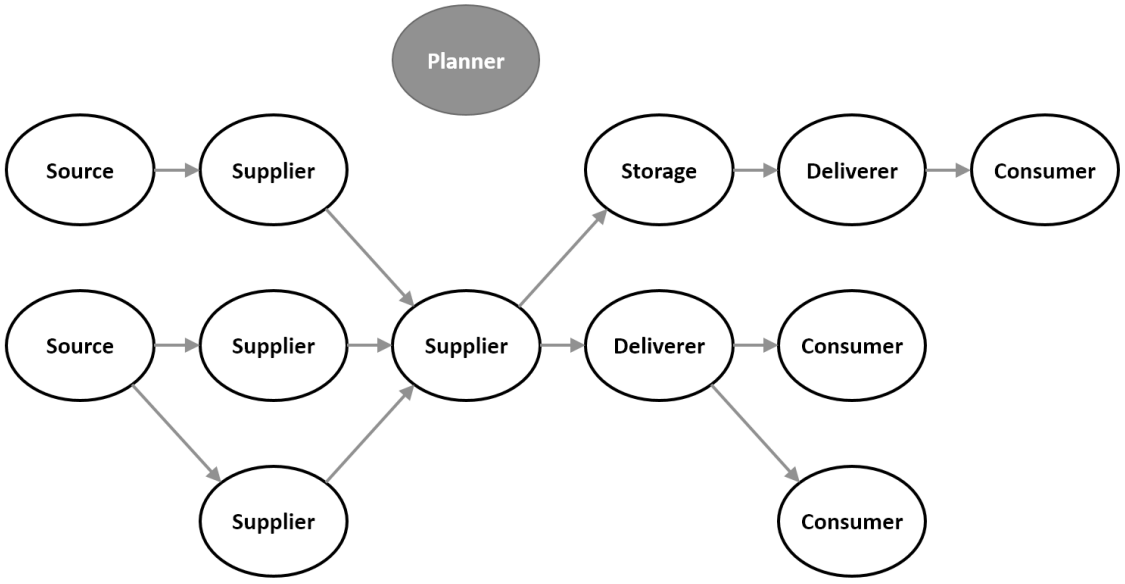


Abbildung 19: Beispiel einer Supply Chain dargestellt in einer abstrakten Notation aus (Umeda und Lee, 2004)

Umeda und Zhang's Modell verfügt über vier Sichten. Die Organisationssicht kategorisiert die an einer Supply Chain beteiligten Partner in sechs Typen. Diese sind „Source", „Supplier", „Storage", „Deliverer", „Consumer" und „Planner"[25]. Diese Typen dienen auch als ein

[25] Im Folgenden behalten wir die englischen Bezeichnungen für die Partnertypen bei

Klassifizierungsmerkmal der von den jeweiligen Organisationstypen ausgeführten Prozesse. Darüber hinaus werden die Prozesse in einer Steuerungssicht, entweder als „bestandsgetrieben“ und „programmgetrieben“[26] typisiert.

Programmgetriebene Steuerung: Die Steuerung der Supply Chain erfolgt zentralisiert. Ein Planner erstellt ein Produktionsprogramm und schreibt dieses fort. Dafür erhält er Bestellungen von Kunden und Information über die aktuellen Lagerbestände, verarbeitet diese und sendet Aufträge an die Supply Chain Partner. Einem effizienten Informationsflusssystem kommt dabei eine hohe Bedeutung für die Flexibilität der Supply Chain zu. Da die Planung zentral erfolgt, kann auf Nachfrageschwankungen schnell reagiert werden. Diese Art der Steuerung wird auch als „push“ bezeichnet (Heiserich et al., 2011, S. 35).

Bestandsgetriebene Steuerung: Diese dezentrale Steuerungsmethode basiert auf der Beobachtung von Lagerbeständen. Der Lieferant frägt regelmäßig den Lagerbestand eines Materialfluss-abwärts gelegenen Lagers ab und startet die Produktion sobald der Bestand die untere Schwelle erreicht hat. Die Anforderungen an die Gestaltung des Informationsflusses und die Informationsverarbeitung sind hierbei nicht so hoch wie in der programmgetriebenen Steuerung, allerdings birgt diese Methode die Gefahr, dass Nachfrageschwankungen erst verzögert zu Produktionsanpassungen bei flussaufwärts gelegenen Partnern führen. Dadurch kann es zu dem sogenannten Bullwhip-Effekt kommen, der in Kapitel 8 „Evaluation“ erläutert wird. Diese Art der Steuerung wird auch als „pull“ bezeichnet.

Die in Tabelle 1 dargestellten Modellierungskonstrukte können dafür genutzt werden, Modelle verschiedener Supply Chain Konfigurationen zu erstellen. Im Folgenden werden anhand der von Umeda und Zhang entwickelten Struktur und Modellierungskonstrukte einige Beispiele für Supply Chain Konfigurationen dargestellt.

[26] Im Englischen „Stock-driven“ und „Schedule-driven“

Tabelle 1: Modellierungskonstrukte, adaptiert von (Umeda und Jain, 2004, S. 12f)

Organisationssicht	Steuerungssicht	Aktivitäten
Source	bestandsgetrieben	Da Source am Beginn des Materialflusses der Supply Chain steht, beobachtet sie den Materialbestand der von ihr belieferten Supplier. Sobald der Bestand unter ein Limit sinkt, sorgt sie autonom dafür, dass der Ziellagerbestand wieder hergestellt wird.
	programmgetrieben	Source erhält vom Planner über den Informationsfluss Materialbestellungen und arbeitet diese gemäß dem, vom Planner vorgegebenen Programm ab.
Supplier	bestandsgetrieben	Supplier beobachtet den Materialbestand eines (anderen) Suppliers und sorgt, sobald dessen Bestand unter ein Limit sinkt, autonom dafür, dass der Ziellagerbestand wieder hergestellt wird.
	programmgetrieben	Supplier erhält Produktionsaufträge vom Planner über das Informationsflusssystem und führt diese aus.
Storage	bestandsgetrieben	Storage erhält über den Materialfluss Materialien von anderen Supply Chain Partnern, lagert diese (zwischen) und füllt autonom die Lager von Suppliern auf.
	programmgetrieben	Storage erhält über den Materialfluss Materialien von anderen Supply Chain Partnern und lagert diese (zwischen). Sobald ein Abruf vom Planner über den Informationsfluss eintrifft, wird das angeforderte Material über den Materialfluss ausgeliefert.
Planner	auftragsgetrieben	Über den Informationsfluss erhält der Planner Kundenaufträge und sendet Lieferaufträge an den Deliverer. Der Planner prognostiziert die zukünftige Nachfrage und schreibt das Produktionsprogramm (MPS - Master Program Schedule) fort. Der Planner pflegt das Produktionsprogramm. Im Falle der bestandsgetriebenen Steuerung setzt der Planner die Wiederauffüllpunkte der einzelnen Storages fest und passt diese gegeben falls der Angebot/Nachfrage Situation an.
Deliverer		Der Deliverer erhält Lieferaufträge von anderen Supply Chain Partnern über den Informationsfluss.
Consumer		Der Consumer erteilt Kaufaufträge an den Planner.

2.3.9.1.1 Programmgetriebene Supply Chain Konfiguration

Abbildung 20 zeigt ein Beispiel für eine Supply Chain, die von einem Produktionsprogramm getrieben wird. Es ist gut ersichtlich wie der Planner im Mittelpunkt des Informationsflusses steht. Er erhält die Bestellung vom Kunden und versendet Produktionsaufträge an verschiedene Supply Chain Partner. Bis auf den Lieferauftrag, der direkt vom Supplier an den Deliverer gesendet wird, laufen alle Informationsflüsse über den zentralen Planner. Der Materialfluss hingegen läuft sequentiell durch die Stufen der Supply Chain, ohne den Planner zu tangieren.

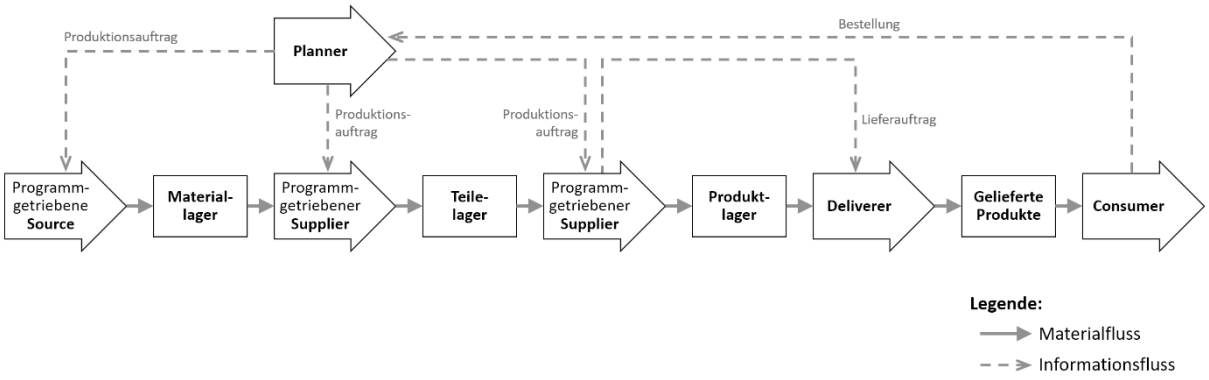


Abbildung 20: Beispiel einer programmgetriebenen Supply Chain, adaptiert aus (Umeda und Jain, 2004)

Abbildung 20 wurde gegenüber dem Original von Umeda und Jain um die Darstellung des Materialflusses erweitert. Sie verzichten in ihrer graphischen Darstellung des Prozesses auf die explizite Visualisierung des Materialflusses und zeigen nur den Informationsfluss mittels Pfeilen an. Das mag für diese einfache sequentielle Wertschöpfungskette ausreichend sein, sobald aber alternative oder parallele Materialflüsse auftreten können, müssen die möglichen Materialflüsse klar definiert werden, um einem Simulationsalgorithmus eindeutige Eingabeparameter zu übergeben. In einer späteren Publikation (Umeda, 2013) stellt Umeda beide Flüsse dar. Ein weiteres Problem dieser Darstellung ist, dass die Auftrittsreihenfolge der einzelnen Informationsflüsse aus der graphischen Darstellung nicht hervorgeht und damit nicht ersichtlich ist, welche Information welche Aktion auslöst. Diese Sequenz und Möglichkeit diese Ursache/Wirkungs-Zusammenhänge zu definieren, wird eine Anforderung in der hier entwickelten Methode werden.

2.3.9.1.2 Bestandsgetriebene Supply Chain Konfiguration

In einer rein bestandsgetriebenen Supply Chain Konfiguration vereinbart der Planner zunächst minimale Lagermengen mit allen Suppliern der Supply Chain, siehe Abbildung 21. Dieser Informationsaustausch erfolgt zu Beginn der Zusammenarbeit und später nur mehr bei Bedarf. In dieser sehr dezentralen Steuerung hat der Planner keinen direkten Einfluss auf

das operative Geschehen in der Supply Chain. Er erhält weiterhin die Bestellinformationen vom Kunden, aber hauptsächlich um die Nachfrageprognose zu erstellen und zu prüfen, ob die minimalen Lagermengen angepasst werden müssen. Die Source und die einzelnen Supplier handeln autonom, indem sie den jeweiligen Ist-Lagerbestand abfragen, mit dem vereinbarten Minimum Lagerbestand vergleichen und falls das Minimum unterschritten ist, sich selbst einen Produktionsauftrag erteilen. Wie viele Einheiten produziert werden wurde zu Beginn mit dem Planer vereinbart.

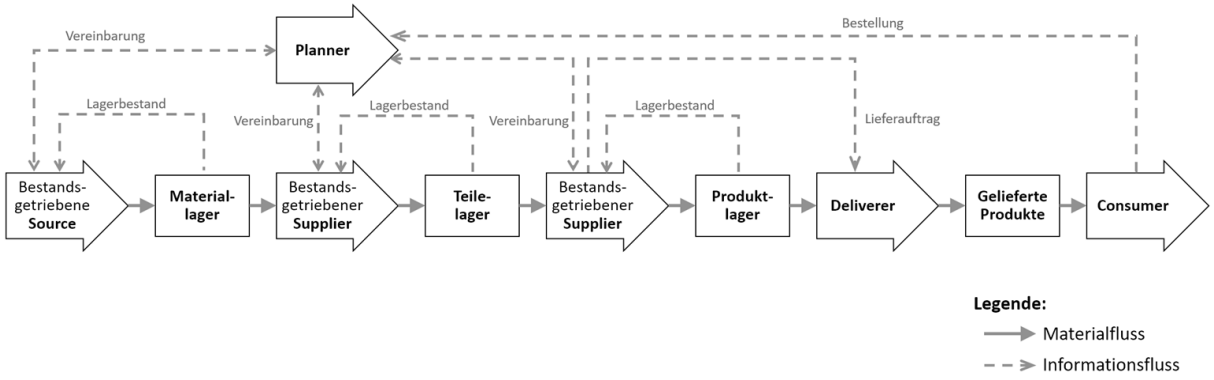


Abbildung 21: Bestandsgetriebene Supply Chain Konfiguration adaptiert aus (Umeda und Jain, 2004)

2.3.9.1.3 Programm- und Bestandsgetriebene Supply Chain Konfiguration

In Abbildung 22 ist eine gemischte programm- und bestandsgetriebene Konfiguration dargestellt. Die Source geht mit dem Planner eine Vereinbarung über den minimalen Materiallagerbestand ein und beobachtet diesen dann. Daraus ergibt sich der zusätzliche Informationsfluss zwischen Materiallager und der bestandsgetriebenen Source. Sobald der Bestand die vereinbarte Grenze unterschreitet, produziert die Source weiteres Material, um den Lagerbestand wieder aufzufüllen.

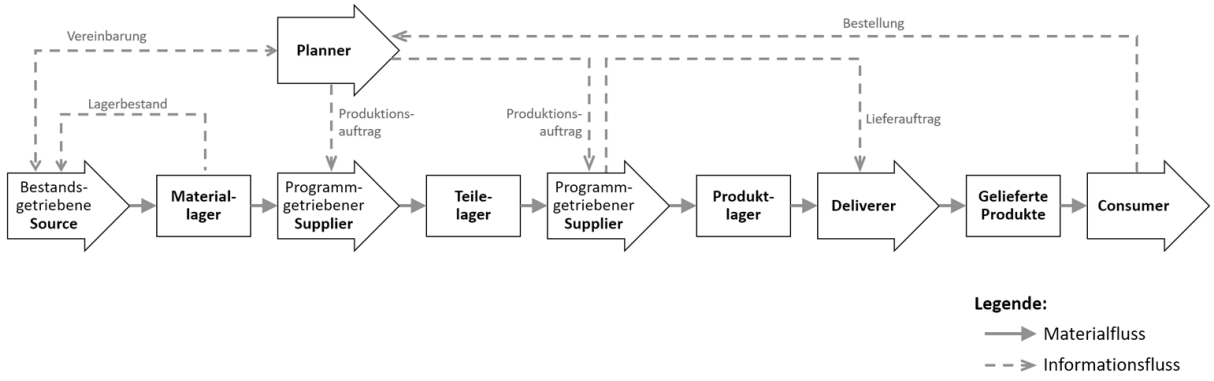


Abbildung 22: Programm- und Bestandsgetriebene Supply Chain Konfiguration adaptiert aus (Umeda und Jain, 2004)

Verändert man die Konfiguration aus Abbildung 22 dermaßen, dass auch der programmgetriebene Teile-Supplier zu einem bestandsgetriebenen Teile-Supplier wird, so zeigt die Konfiguration eine typische Konfiguration für einen Assembler, der erst bei Vorliegen einer tatsächlichen Kundenbestellung das Produkt aus Teilen zusammenbaut, die auf Lager liegen.

2.4 Instandhaltung

Ein Erfolgsfaktor von im Wettbewerb stehenden Organisationen und deren Supply Chains, ist die Fähigkeit Rationalisierungspotentiale zu identifizieren und auszuschöpfen. Eine Mittel zur Reduktion der operativen Kosten, im Speziellen durch Senkung der Personalkosten, ist die Erhöhung des Automatisierungsgrades von Produktions- und Logistikprozessen. Dies inkludiert nicht nur die direkt mit der Produktion befassten Mitarbeiter, sondern auch indirekte Positionen, wie beispielsweise Planung und Disposition, welche durch entsprechende Softwaresysteme unterstützt oder ersetzt werden. Mit steigender Automation steigt die Notwendigkeit die Zuverlässigkeit und Verfügbarkeit von Produktions- und Transportanlagen sicherzustellen, dies ist der Hauptzweck aller Instandhaltungstätigkeiten. Sie sorgen für eine gleichbleibende Produktqualität und begrenzen das Risiko und die Folgekosten von Produktionsausfällen.

Im Folgenden werden die Instandhaltung und ihre Komponenten definiert.

2.4.1 Begriffe der Instandhaltung

Zur Definition der in dieser Arbeit verwendeten Begriffe rund um die Instandhaltung werden die Normen des Deutschen Instituts für Normung e. V. herangezogen. Grundmaßnahmen und Struktur der Instandhaltung werden für die deutsche Sprache in der DIN 31051 (DIN Deutsches Institut für Normung e. V., 2011) definiert. DIN EN 13306 (DIN Deutsches Institut für Normung e. V., 2010) definiert englische Begriffe der Instandhaltung[27] die allerdings nicht deckungsgleich mit den deutschen Begriffen aus DIN 31051 sind. Vielmehr werden in DIN EN 13306 verschiedenen Instandhaltungsarten und das Instandhaltungsmanagement definiert, siehe Abbildung 23.

Die DIN 31051 definiert Instandhaltung als

„Kombination aller technischen und administrativen Maßnahmen sowie Maßnahmen des Managements während des Lebenszyklus einer Einheit, die dem Erhalt oder der Wiederherstellung ihres

[27] Im Englischen „Maintenance“

funktionsfähigen Zustands dient, sodass sie die geforderte Funktion erfüllen kann.“ (DIN Deutsches Institut für Normung e. V., 2011)

Ein in dieser Definition verwendeter und für diese Arbeit wichtiger Begriff ist die „Einheit“, welche die Norm definiert als

„Teil, Bauelement, Gerät, Teilsystem, Funktionseinheit, Betriebsmittel oder System, das/die für sich allein beschrieben und betrachtet werden kann“ (DIN Deutsches Institut für Normung e. V., 2011)[28]

Ein Ersatzteil ist eine

„Einheit zum Ersatz einer entsprechenden Einheit, um die ursprünglich geforderte Funktion der Einheit zu erhalten“

Wie aus Abbildung 23 ersichtlich, besteht die Instandhaltung aus den Grundmaßnahmen Wartung, Inspektion, Instandsetzung und Verbesserung.

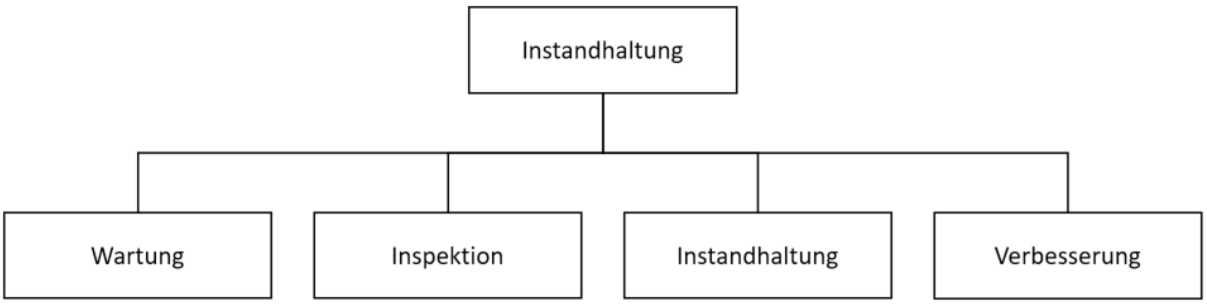


Abbildung 23: Struktur der Instandhaltung nach DIN 31051

2.4.1.1 Wartung

Wartung umfasst alle Aktivitäten, die zu einer Verzögerung des vorhandenen Abnutzungsvorrats der Einheit führen.

Jede Einheit unterliegt einer Abnutzung, sei es durch physikalische oder chemische Vorgänge. Der Abnutzungsvorrat ist hierbei der Vorrat der möglichen Funktionserfüllungen unter festgelegten Bedingungen, der einer Einheit innewohnt. Je nach Instandhaltungsmaßnahme kann dieser Abnutzungsvorrat wieder teilweise oder zur Gänze aufgefüllt werden, siehe Abbildung 24.

[28] Bemerkung: im Englischen wird eine Einheit als „Maintainable Unit“ bezeichnet, diese wird als kleinste instand zu haltende Komponente einer Anlage angesehen. Bei einem Austausch wird beispielsweise die gesamte „Maintainable Unit“ ersetzt

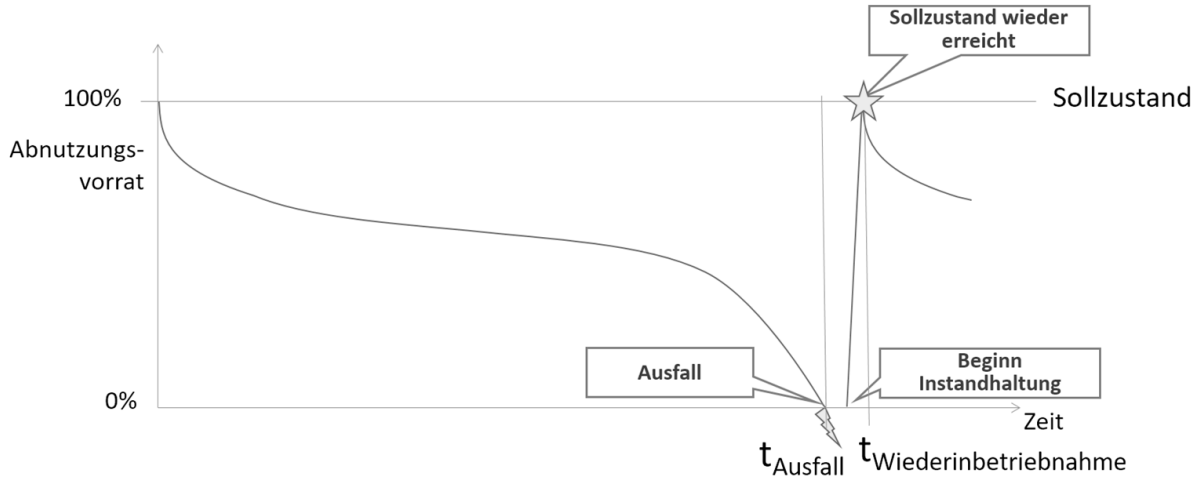


Abbildung 24: Beispielhafte Abbaukurve des Abnutzungsvorrats

2.4.1.2 Inspektion

Die Inspektion umfasst alle Tätigkeiten, die zur Feststellung und Beurteilung des aktuellen Zustandes einer Einheit dienen. Des Weiteren sind Maßnahmen enthalten, die zur Ursachenidentifikation der Abnutzung führen, wie zum Beispiel Fehleranalyseverfahren (Failure mode and effect analysis - FMEA) und die zur Bestimmung der Lösung führen. Die Inspektion kann periodisch erfolgen, durch ein Ereignis ausgelöst werden oder permanent als Zustandsmonitoring durchgeführt werden.

2.4.1.3 Instandsetzung

Die Instandsetzung umfasst alle physischen Tätigkeiten die zu der Wiederherstellung der Funktion einer fehlerhaften Einheit führen, also zum Beispiel Austauschen oder Ausbessern von Einheiten und deren Funktionsprüfung.

Instandsetzungsmaßnahmen sind nicht in einem längerfristigen Plan enthalten, da sie Konsequenz eines Ausfalls sind, der eigentlich durch eine, in einer Instandhaltungsstrategie enthaltenen, Wartung der Einheit verhindert werden soll.

Die Vorbereitung und Durchführung der Instandsetzung ist mit mehr Aufwand und Kosten verbunden als die der Wartung und Inspektion (Bosch, 2016) da Materialien, Personal und Hilfsmittel meist unter Zeitdruck, da der Betriebsablauf gestört wird, bereitgestellt werden müssen.

2.4.1.4 Verbesserung

Die Instandhaltungs-Grundmaßnahme Verbesserung umfasst alle technischen und administrativen Maßnahmen, die zur Steigerung der Zuverlässigkeit, der Instandhaltbarkeit und Sicherheit von Einheiten führen, ohne dass ihre ursprüngliche Funktion geändert wird.

Instandhaltung beschränkt sich nicht nur auf die Durchführung der technischen Maßnahmen, sondern umfasst auch die Planung, Beauftragung, Information, Dokumentation, Bewertung der Risiken bezüglich Sicherheit und der Bereitstellung und Erhalt von Wissens[29].

Wie aus Abbildung 25 ersichtlich, unterscheidet DIN EN 13306 zwischen den Instandhaltungsarten Verbesserungen und Präventiver Instandhaltung, die zum Ziel haben Fehler zu vermeiden und Korrektiver Instandhaltung, die nach Eintreten eines Fehlers durchgeführt wird.

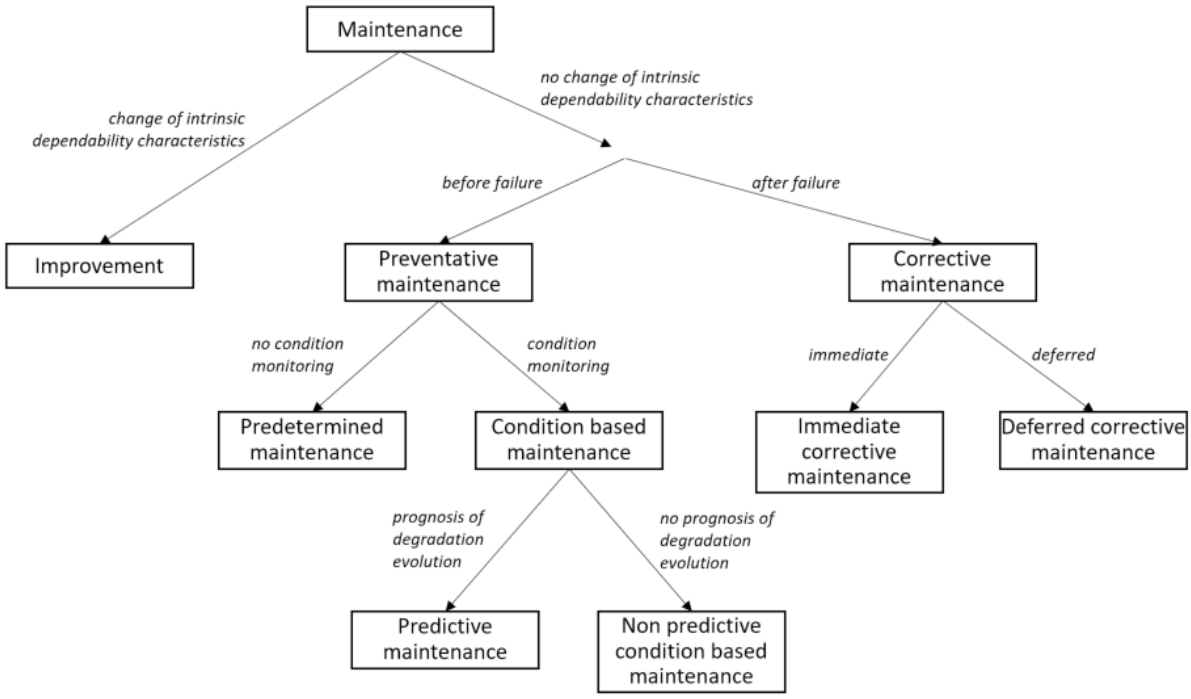


Abbildung 25: Instandhaltungsarten nach DIN EN 13306 (Bosch, 2016)

2.4.1.5 Total Productive Maintenance

TPM oder Total Productive Maintenance (Nakajima, 1988) und (Ahuja und Khamba, 2008) ist ein Ansatz der erstmals bei der Nippon Denso Co. Ltd., einem Zulieferer der Toyota Motor Company, im Jahr 1971 eingeführt wurde und seither über die eigentliche Instandhaltung

[29] Diese Themen werden beispielsweise im EU-Projekt ComVantage behandelt. Siehe http://comvantage.eu Zugriff am 25.03.2017

hinweg im gesamten Produktionssystem angewendet werden kann[30]. Ähnlich wie Lean Production (Krafcik, 1988), (Shah und Ward, 2007) und Kaizen (Paul Brunet und New, 2003) handelt es sich um ein kontinuierliches Verbesserungsprogramm, welches das gesamte Unternehmen miteinschließt.

Zielsetzung der Anwendung von TPM ist, durch eine effizientere Nutzung der Produktionsanlagen die Produktivität des Unternehmens zu erhöhen. Dafür werden alle Mitarbeiter, die direkt oder indirekt an den Produktionsanlagen arbeiten, involviert. Dazu zählt man nicht nur Bediener und Instandhaltungspersonal, sondern auch Ingenieure die mit der Konstruktion der Anlage betraut waren. Diese sind beispielsweise Qualitätsmanager, Einkäufer und Betriebsorganisatoren (Kuhn et al., 2006, S. 28). Während der ersten Betriebsjahre der Anlage wird Wissen über die Anlage, wie beispielsweise Schwachstellen, Zuverlässigkeit, und Indikatoren für Fehler, schrittweise aufgebaut und vertieft. Die Mitarbeiter arbeiten als Team, mit dem Ziel, den Bedienern genug Fähigkeiten zu vermitteln, damit sie gewisse Instandhaltungstätigkeiten, wie einfache Wartungsmaßnahmen oder Inspektionen, autonom durchführen können. Die Einbeziehung von Konstrukteuren und Einkäufern hat zum Ziel aus den gewonnenen Erkenntnissen zu lernen und diese in Zukunft anzuwenden.

2.4.1.6 e-Maintenance

E-Maintenance ist ein relativ neues Instandhaltungskonzept das mit dem Durchbruch von Technologien wie Internet of Things (IoT), RFID und mobile Endgeräte, möglich wurde. Obwohl die Bedeutung des Buchstaben „e“ von manchen Autoren mit „excellent“, „efficient“, „effective“ oder „enterprise“ angegeben wird (Baldwin, 2000) werden unter e-Maintenance im Allgemeinen und in dieser Arbeit, Instandhaltungsverfahren, mit denen der Zustand von Anlagen und ihre Einheiten mit Hilfe von Informations- und Kommunikationstechnologien überwacht und gemanagt wird, verstanden.

Entsprechend dieser Definition ist e-Maintenance sehr breit gefasst und es gibt eine große Anzahl diesbezüglicher Publikationen quer durch die Domäne der Instandhaltung. Koç (Koç und Ni, 2005) ordnet e-Maintenance als ein Bestandteil des e-Manufacturing ein, was wiederrum ein Teil eines e-Business sein kann, siehe Abbildung 26. Die Umsetzung von e-Maintenance wird im Allgemeinen die Informationslogistik, also den Informationsfluss der Supply Chain maßgeblich verbessern (Karim et al., 2009) und kann zu einer Effizienzsteigerung der administrativen Geschäftsprozesse führen (Hausladen und Bechheim, 2004). Durch die Möglichkeit, permanent Daten über den Anlagenzustand,

[30] Dementsprechend wird TPM auch häufig als Total Productive Manufacturing oder Total Productive Management interpretiert

Anlagenperformanz und Produktqualität zu sammeln, können Erkenntnisse gewonnen werden, die zu einer besseren Prognose von notwendigen Instandhaltungsmaßnahmen führen können (Lee et al., 2006).

Das EU-Projekt ComVantage, das in dieser Arbeit im Kapitel 2.4.2.1 kurz beschrieben wird, behandelt mit dem Anwendungsfall der mobilen Instandhaltung einen Teilbereich der e-Maintenance.

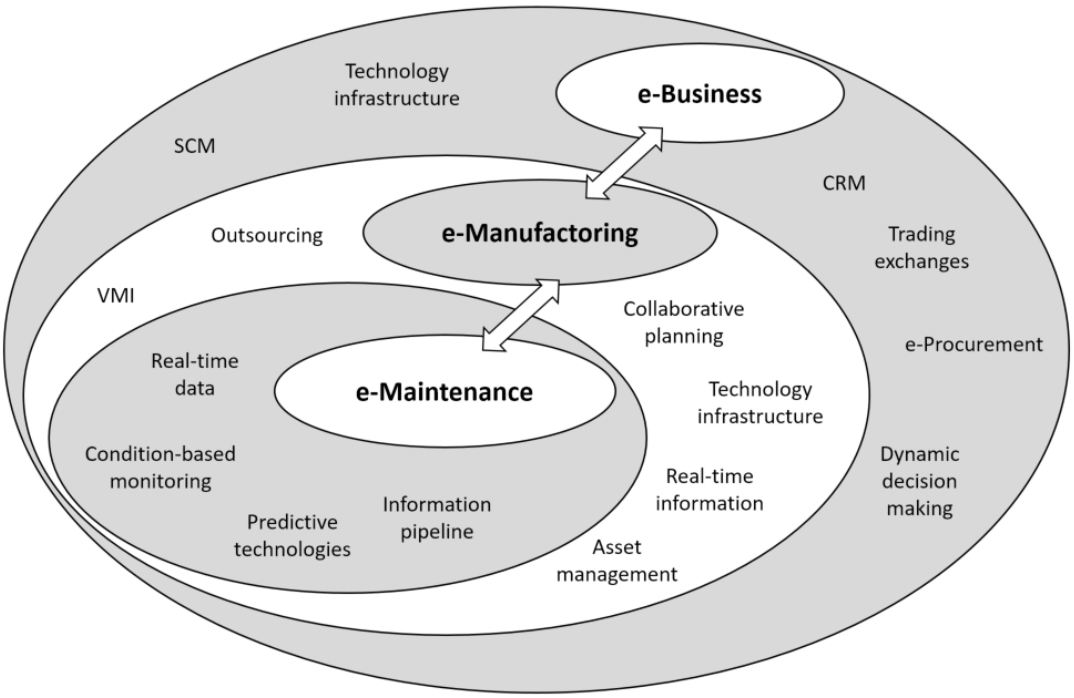


Abbildung 26: Integration von e-Manufacturing mit e-Maintenance und e-Business (Koç und Ni, 2005)

Einen guten Literaturüberblick findet sich in (Muller et al., 2008) beziehungsweise (Levrat et al., 2008)

2.4.2 Instandhaltungs-/ Maintenance Supply Chain

Die Instandhaltungs-/Maintenance Supply Chain ist eine Spezialisierung der Supply Chain. Ihr Hauptaugenmerk liegt in der zeitgerechten und kostenadäquaten Bereitstellung von Ersatzteilen, Information, Werkzeugen, Anweisungen und Personal zur Instandhaltung von Anlagen.[31] Der Kunde ist somit der Betreiber der Anlagen. Die Maintenance Supply Chain kann die gesamte Instandhaltungsdienstleistung übernehmen oder nur Teilaspekte davon abdecken. Die Spare Part Supply Chain oder Ersatzteil Supply Chain ist ein Untertyp der

[31] Die Begriffe „Maintenance Supply Chain“ und „Instandhaltungs-Supply Chain“ sind deckungsgleich, in der englischsprachigen Literatur wird auch häufig von MRO (Maintenance, Repair, Overhaul) Supply Chains gesprochen.

Maintenance Supply Chain, der sich nur mit der Bereitstellung der für die Instandhaltung benötigten Ersatzteile befasst.

Instandhaltungsdienstleister agieren in komplexen Supply Chains mit Partnern wie beispielsweise Ersatzteillieferanten, Logistikunternehmen und Servicedienstleister, die spezialisierte Instandhaltungstätigkeiten anbieten oder Arbeitsmittel vermieten. Diese Supply Chains sind von starken internen Abhängigkeiten geprägt. Dies setzt eine starke Verzahnung und Synchronisation der Arbeitsabläufe und den Informationsaustausch voraus, um dem Kunden die zugesicherte Verfügbarkeit seiner Anlagen zu garantieren (Fabry und Schmitz-Urban, 2010).

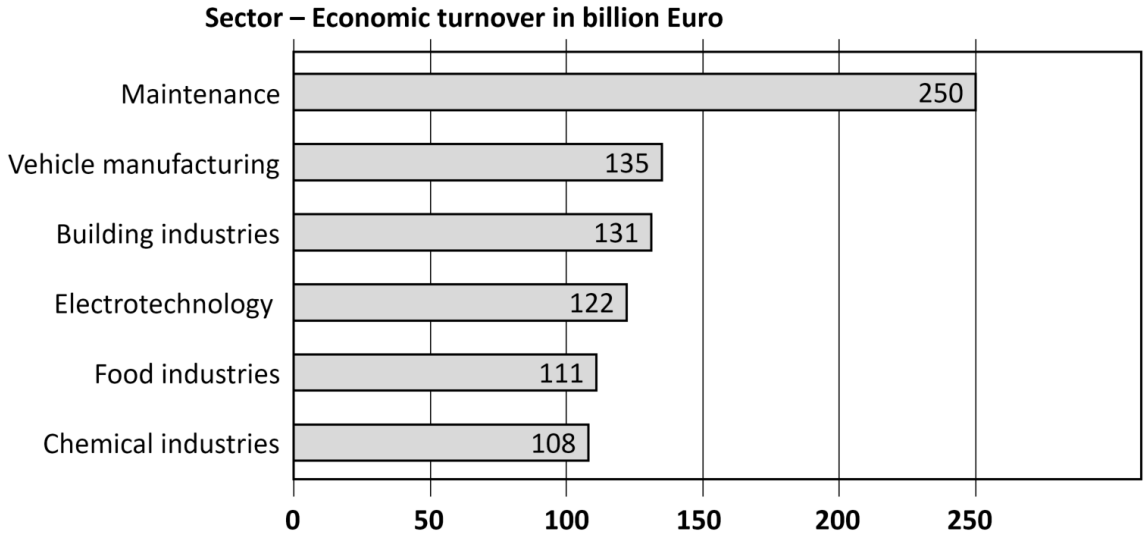


Abbildung 27: Wirtschaftliche Bedeutung des Instandhaltungssektors in Deutschland (Fabry und Schmitz-Urban, 2010).

Eine weitere wirtschaftliche Motivation ist, dass die Instandhaltungskosten einer produzierenden Anlage im Vergleich zu den indirekten Kosten eines Anlagenausfalls, verursacht zum Beispiel durch Produktionsausfall und erhöhte Kosten der ungeplanten Reparatur, im Verhältnis 1: 5 stehen (Fabry und Schmitz-Urban, 2010). Auch Huiskonen (Huiskonen, 2001, S. 127) argumentiert, dass die Kosten für Ersatzteile und deren Einbau viel geringer sind als die monetären Auswirkungen eines in Beziehung stehenden Fehlers. Herausforderung gerade im Bereich des Ersatzteilmanagements ist es, den optimalen Lagerbestand und Lagerort pro Ersatzteil zu ermitteln, um schnellstmöglich auf Störungen reagieren zu können. Es gibt viele in der Literatur beschriebene Ansätze, um dieses Problem mittels mathematischer Modelle zu optimieren. Gemein ist allen Modellen, dass sie für einen Nicht-Mathematiker schwer zu verstehen sind und nur auf speziell eine Fallkonstellation anwendbar sind (Muckstadt, 2005), (Molenaers et al., 2011), (Kennedy et al., 2002).

Abbildung 28 zeigt die typische Zusammensetzung der Durchlaufzeit von der Identifikation eines Fehlers bis zur Behebung dieses.

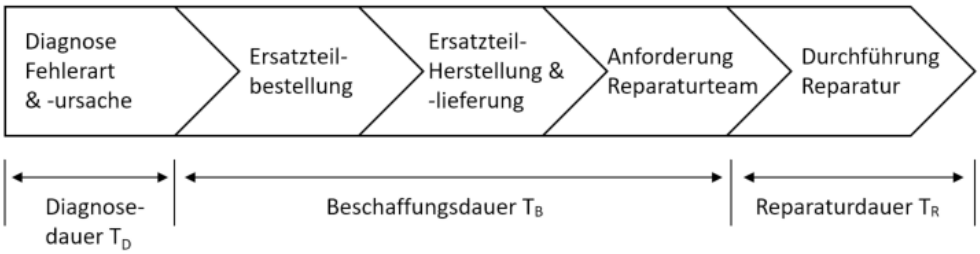


Abbildung 28: Zeitlicher Ablauf der Instandsetzung (Trost, 2008)

Im Folgenden werden die in Kapitel 2.3.3 beschriebenen Treiber des Supply Chain Managements auf die Spezialisierung der Maintenance Supply Chain angewendet.

2.4.2.1 Veränderung der Märkte durch Informations- und Kommunikationstechnologie (Geschwindigkeit)

Informations- und Kommunikationstechnologie erhöht die Transparenz, sowohl auf Angebots- als auch auf Nachfrageseite. Zum einen hat der Kunde durch Verwendung von Such- und Preisvergleichsportalen im Internet die Möglichkeit sich einen guten Überblick über Preise und Leistungen zu verschaffen, zum anderen bekommt der Anbieter die Möglichkeit über Datenanalyse (Big Data) seine Interessenten und Kunden besser kennenzulernen, ihr Kaufverhalten zu verstehen und dieses im gewissen Maße zu beeinflussen. Ein kollaborativer Ansatz bei der Informationsweitergabe vorausgesetzt, kann die Informationstechnologie auch für eine schnelle Weitergabe von Verkaufsinformationen an vorgelagerte Supply Chain Partner sorgen. Sieht man als „Kunden“ der Maintenance Supply Chain die instand zu haltende Einheiten an, so lässt sich diese Erkenntnis auch auf die Instandhaltung anwenden. Der Einsatz von Computerized Maintenance Management Systems (CMMS)[32], mobilen Endgeräten und smarte Sensorik führt zu einer erhöhten Transparenz über den Zustand und somit die Instandhaltungserfordernisse der Anlagen und erlauben eine bessere Planung und effizientere Durchführung der notwendigen Tätigkeiten. Die Fähigkeit der Planung, der frühzeitigen, bedarfsorientierten Bestellung von Ersatzteilen und speziellen Services und der Einplanung von Instandhaltungstätigkeiten gemäß bester Ressourcenverfügbarkeit, hilft die Effizienz der Instandhaltung zu erhöhen.

Die durchgehende Verwendung von Informationstechnologie mit definierten Übergabeformaten und Schnittstellen verringert, durch Wegfall manueller Schritte, die Gefahr von fehlerhaften Informationsweitergaben, wie zum Beispiel Ersatzteilnummern, Beginn- und Endzeiten von Wartungsaktivitäten, Zustandsinformation und Lokation von zu wartenden Anlagen und beschleunigt die Information aller beteiligter Supply Chain Akteure. Dies

[32] Auch als Computerized Maintenance Management Information Systems (CMMIS) bezeichnet

wiederrum gibt jedem einzelnen Akteur in der Wertschöpfungskette die Möglichkeit einer flexibleren Planung und vergrößert Pufferzeiten. Ist beispielsweise ein Ersatzteil beim Ersatzteilelieferanten nicht auf Lager, so kann noch genügend Zeit zur Verfügung stehen, den Produzenten des Ersatzteils zu beauftragen.

Ein Informationssystem trägt auch dazu bei, einen besseren Überblick über alle anstehenden Instandhaltungstätigkeiten zu gewinnen und kann so zu Kosteneinsparungen oder Effizienzgewinnen durch Bündelung ähnlicher Tätigkeiten beitragen.

Das EU-Forschungsprojekt ComVantage[33] untersucht Möglichkeiten der Kollaboration unter Unternehmen die sich durch mobile Applikationen und der Verwendung von ‚Linked Data' (Bizer et al., 2009), (Open Models Laboratory, 2014, S. 13) ergeben. Ein Forschungsgegenstand ist dabei das Applikationsfeld der mobilen Instandhaltung und die Kommunikation von Sensoren, die als Teil des Internet of Things (IoT) direkt mit Instandhaltungskoordinatoren kommunizieren (Buchmann, 2014).

ComVantage benutzt den Ausdruck des „Virtuellen Unternehmens"[34] das nach Definition von (Barnett et al., 1994) ein Zusammenschluss mehrerer eigenständiger Unternehmen für einen begrenzten Zeitraum ist. Diese Allianz bildet sich, um einen bestimmten Zweck zu erfüllen. Das virtuelle Unternehmen an sich hat keine eigenen bedeutenden Ressourcen, sondern bedient sich der Ressourcen der Partnerunternehmen. Auch wenn der temporäre Charakter nicht immer gegeben ist, kann eine Supply Chain als Virtuelles Unternehmen angesehen werden, das von eigenständigen Supply Chain Partnern gebildet wird.

Im EU Projekt ComVantage wurden sechs zentrale Anforderungen identifiziert und adressiert (Münch et al., 2014):

Virtualität: Vermeidung einer zentralen Steuerungs- und Datenhaltungskomponente. Prozesse und deren Ressourcen und Daten sind ausschließlich dezentral bei den Partnern angesiedelt.

Datensicherheit: Es muss sichergestellt werden, dass ausschließlich befugte Partnerunternehmen Zugriff auf Daten haben.

Flexibilität: Virtualität erleichtert flexible Prozesse, darüber hinaus sind jedoch konfigurierbare Plattformen, wie Workflowsysteme, und generische Komponenten, wie standardisierte Datenschnittstellen, notwendig, um anwendungsspezifische Anpassungen zu ermöglichen.

[33] Siehe http://comvantage.eu

[34] Englisch Virtual Enterprise

Kollaboration: Mobile Apps unterstützen die Ausführung von unternehmensübergreifenden Workflows und stellen arbeitsschrittrelevante Informationen dem Bearbeiter zur Verfügung.

Mobilität: die Applikationen sollen auf gängigen, kostengünstigen mobilen Endgeräten lauffähig sein.

Gebrauchstauglichkeit: Neben einer benutzerfreundlichen Bedienungsoberfläche müssen die Endgeräte robust genug gebaut werden, um zuverlässig den rauen Alltagsbedingungen des Instandhaltungsbetriebs zu überstehen.

2.4.2.2 Wechsel von Push- zu Pull Märkten

Die fortschreitende Einführung von zustandsabhängiger Instandhaltung anstelle der periodisch, vorbeugenden (Schuh et al., 2005) kann mit einem Übergang von push zu pull gleichgesetzt werden. Anstatt von fixen Wartungsintervallen, in denen Teile gewartet und getauscht werden, basiert die zustandsabhängige Wartung auf regelmäßigen, geplanten Inspektionen oder permanenter Zustandsüberwachung[35]. Je nach Ergebnis der Inspektion wird eine Wartung eingeplant oder auf den nächsten Zyklus verschoben. Der Bedarf an Ersatzteilen ist somit stärker an die Abnutzung der Anlagen gekoppelt und somit für einen Außenstehenden nicht mehr einfach zu prognostizieren. Damit steigt die Gefahr einer Unterdeckung von Ersatzteilen beim Lieferanten (Saalmann et al., 2016).

Der Pull-Effekt ist noch ausgeprägter, wenn die Anlage über eigene Sensorik verfügt und autonom ihre Wartung über das IoT und CMMS beauftragen kann. Typische Sensoren messen Schwingungen, Temperatur, Drehmoment, Stromaufnahme, Schmierstoffzusammensetzung oder analysieren die Qualität des gefertigten Produkts. Eine Möglichkeit einen Ausfall wegen Nichtverfügbarkeit von Ersatzteilen zu umgehen ist, auf Basis der Nutzungsdaten der Anlage die Wartungsbedarfe zu prognostizieren[36] und diese Information an den Instandhaltungsdienstleister und Ersatzteillieferanten weiterzuleiten (Espíndola et al., 2012). Eine Synchronisation der Material- und Informationsflüsse ist ein Erfolgsfaktor für diesen kollaborativen Ansatz. Er führt zu einem reduzierten Ersatzteillagerbedarf, stellt sicher, dass dem Instandhaltungspersonal alle notwenigen Teile zur Durchführung der Tätigkeit zur Verfügung stehen und erlaubt somit eine effizientere Spare Part Supply Chain (Fabry und Schmitz-Urban, 2010).

[35] Englisch „Condition Monitoring“

[36] Englisch „Predictive Maintenance”

2.4.2.3 Steigende Komplexität

Mit fortschreitender Komplexität der Anlagen und der immer stärkeren Verzahnung von Hardware und steuernder Software werden auch die Anforderungen an die Instandhaltungsingenieure immer höher. Der Bedarf an Spezialisten für der Instandhaltung von Anlagen steigt (Meixell et al., 2008), was zum einen eine Herausforderung für die Identifikation der benötigten Ressourcen für eine Instandhaltungstätigkeit darstellt und zum anderen ein generelles Planungsproblem für den optimierten Einsatz dieser spezialisierten Ressourcen bedeutet. Neben Mitarbeiterressourcen müssen für die Instandhaltung komplexerer Anlagen auch die geeigneten Werkzeuge zur Verfügung gestellt werden, die notwendigen Genehmigungen beschafft oder kontrolliert werden, alle am Prozess beteiligten Stellen informiert und die Absicherungen der Arbeitsstelle organisiert werden. Des Weiteren muss sichergestellt werden, dass die richtigen Ersatzteile zur Verfügung stehen oder nachdem das Problem identifiziert wurde, diese schnellstmöglich organisiert werden können, um die Maschinenausfallzeit zu minimieren (Saalmann et al., 2016). Fehlende Ersatzteile können zur Beschädigung weiterer Komponenten der Maschine führen oder die Produktqualität verschlechtern (Meier und Klimek, 2008).

Durch die immer flachere Fertigungstiefe, die zu einer hohen Anzahl von Lieferanten für Anlagenkomponenten führen kann, wird auch die Spare Part Supply Chain komplexer. Sind diese Lieferanten global verteilt, müssen die daraus resultierenden Lieferzeiten, als Entscheidungskriterium für eine Vor-Ort-Lagerhaltung in Betracht gezogen werden. Des Weiteren erschwert eine kundenindividuelle Fertigung die Abschätzung von Ersatzteilbedarfen. Diese Komplexität in materialflussrelevanten Aspekten der Supply Chain wird durch die notwenige Interaktion und Informationsweitergaben noch verstärkt. Eine Information aller relevanten Supply Chain Partner führt zwar, wie zuvor gezeigt, im Allgemeinen zu einer effizienteren Bereitstellung von Produkten, bedarf aber auf der anderen Seite die Anpassung und Synchronisation von Informationsflüssen mit je nach Akteur unterschiedlicher Syntax und Semantik.

2.4.2.4 Die Rolle der Dynamik

Wie zuvor erläutert, kann der Bedarf für Ersatzteile sehr stark schwanken und schwer zu prognostizieren sein (Huiskonen, 2001), (Hellingrath und Cordes, 2014). Im Falle von ungeplanten Maschinenausfällen werden Ersatzteile sofort benötigt und erfordern daher eine Supply Chain, die eine sehr schnelle Reaktion auf diese Ereignisse zulässt. Während Phasen der geplanten, periodisch vorbeugenden Instandhaltung ist hingegen eine effiziente Ressourcennutzung, also beispielsweise ein geringer Lagerbestand von Ersatzteilen, optimierte Transportwege und bestmögliche Ausnutzung von Personalressourcen anzustreben. Dies zeigt, dass die Maintenance Supply Chain sehr flexibel gestaltet werden

muss, um der zu erwartenden Dynamik im Instandhaltungsgeschäft gerecht zu werden und somit die Koordination der zugrundeliegenden Informations- und Materialflüsse ein wesentlicher Erfolgsfaktor ist. Ein Beispiel für den Informationsfluss in einer Maintenance Supply Chain ist in Abbildung 11 dargestellt.

2.5 Simulation

Ziel der Modellbildung und der anschließenden Simulation ist es, das Verhalten von dynamischen Systemen, wie es Supply Chains sind, nachzubilden und deren Verhalten mit vertretbarem Aufwand vorherzusagen. In der Literatur sind eine Vielzahl von Definitionen zum Simulationsbegriff zu finden, eine tiefergehende Diskussion zum Begriff der Simulation bietet Frank in (Biethahn et al., 2013). Für diese Arbeit ist vor allem relevant, dass die Simulation:

- auf einem Modell der Realität durchgeführt wird
- computerbasiert erfolgt und wiederholbar ist
- das Verhalten und die Zustandsänderung des Modells im Zeitverlauf abbildet und als Ergebnis zur Verfügung stellt. Banks (Banks, 1999) spricht von einer „artificial history of the system“ die von der Simulation erzeugt wird
- Veränderung der Eingangsparameter und damit das Experimentieren mit Szenarien erlaubt
- für die Anwendungsdomäne nachvollziehbare Ergebnisse liefert, die zum besseren Verständnis der Zusammenhänge führen und damit Verbesserungspotentiale erkennen und bewerten lassen

Unter dieser Betrachtung ist die folgende, von Robinson entwickelte Definition für Simulation passend (Robinson, 2014)

„Experimentation with a simplified imitation (on a computer) of an operations system as it progresses through time, for the purpose of better understanding and/or improving that system."

Wobei Robinson in seiner Definition das Modell als “simplified imitation (on a computer) of an operations system” bezeichnet.

Kleijnen (Kleijnen, 2005) sieht das Anwendungsgebiet der Simulation im Supply Chain Management im strategischen Bereich, inklusive der Neuausrichtung der Supply Chain und dem operativen Bereich zur Analyse der Steuerungsparameter. Er stellt dabei vier verschiede Arten von Simulationsverfahren vor (Kleijnen und Smits, 2003):

- Simulation in einer Tabellenkalkulations-Software („Spreadsheet Simulation“)

- System Dynamics (SD)[37]
- Diskrete Ereignis Simulation[38, 39]
- Business Games[40]

Im Folgenden wird dargestellt, warum die Simulation als Analysemethode für die, in dieser Arbeit entwickelte Methode gewählt wurde. Zunächst werden die Merkmale der Domäne herausgestellt die den Ansatz der Simulation notwendig machen, dann werden Vorteile des Simulationsansatzes aufgelistet und anschließend auf die Nachteile eingegangen.

2.5.1 Notwendigkeit für Simulation von Supply Chain Netzwerken

Schon in relativ einfachen Systemen kann es sich als sehr schwierig herausstellen, ohne adäquate Werkzeuge, eine Reaktion des Systems im Zeitverlauf zu prognostizieren (Bossel, 2004). Das Problem wird mit einer Erhöhung der Vernetzung, durch Hinzufügen von Interaktionsmöglichkeiten zwischen Bearbeitungsstationen oder Supply Chain Partnern noch potenziert.

Nach Umeda und Jain (Umeda und Jain, 2004, S. 8), ist Simulation ein sehr vielfältig einsetzbares Analyseinstrument, das speziell die Untersuchung von Modellen erlaubt, welche Materialflüsse und/oder Informationsflüsse enthalten. In Tabelle 2 werden verschiedene Planungs- und Analyseverfahren verglichen.

[37] Owen (Owen, 2013) vergleicht in seiner Arbeit die Eignung von System Dynamics, Discrete Event Simulation und Agent Based Modelling zur Verbesserung der Leistungsfähigkeit von Supply Chains

[38] Auch als „diskrete ereignisorientierte Simulation „oder „diskrete ereignisgesteuerte Simulation“ bezeichnet.

[39] Zu einem Vergleich von System Dynamics mit Diskreter Ereignis Simulation siehe Tako und Robinson (Tako und Robinson, 2012)

[40] Vergleiche auch Umeda (Umeda, 2010, S. 8f)

Tabelle 2: Methoden zur Planung und Analyse von Produktions- und Supply Chain Systemen, mit leichten Änderungen übernommen aus (Košturiak und Gregor, 1995)

Methode	Beschreibung	Anwendungen
Binäre Logik	Berechnung von binären Zuständen, zum Beispiel intakt / defekt mittels Wahrscheinlichkeitsrechnung	Zuverlässigkeits- und Verfügbarkeitsrechnung von einzelnen Aggregaten, Produktionsprozessen, Lieferketten
Warteschlangentheorie	Ermittlung von Wartezeiten, Service Level Einhaltung, Länge von Warteschlangen, Abfertigungsrate von Engpässen	Lösung von Teilproblemen, bei denen Engpässe, Maschinen oder Wegstecken mit Poisson-Verteilungen als Eingangsgrößen beschrieben werden können
Markov-Ketten	Berechnung von Zustandswahrscheinlichkeiten aus exponentiell verteilten Zustandsübertragungsraten	Verfügbarkeitsrechnung für einfache Produktions- oder Liefersysteme (zum Beispiel zwei Maschinen, ein Lager)
Lineare Optimierung	Bestimmung von Maxima oder Minima einer linearen Zielfunktion, deren Nebenbedingungen linear beschrieben sind	Zuweisungsprobleme, Standortprobleme, Transportprobleme
Dynamische Optimierung	Erweiterung der linearen Optimierung, wobei Zielfunktion und Nebenbedingungen nicht mehr linear sind	Produktionsprogrammplanung, Splitten von Aufträgen
Heuristische Methoden	Bei hoher Komplexität mathematischer Optimierungsaufgaben liefern heuristische Methoden genügend genaue Ergebnisse	Layout Planung, Standortplanung, Reihenfolgeprobleme
Entscheidungsbaum	Gezielte Absuche von Entscheidungsbäumen mit Ausschlüssen von Verzweigungen, die den Zielzustand nicht erhalten	Ziel-Wege-Probleme, Bearbeitungsfolgen, Betriebsmittelanforderung
Prognoserechnung	Extrapolation von Vergangenheitswerten für die Zukunft	Lagerhaltung, Bedarfsermittlung
Simulation	Empirisches Verfahren durch Abbildung des realen Problems in ein Modell und durch Experimente am Modell, um gute Problemlösung zu erhalten	Planung verketteter Anlagen und Supply Chain Partner, Materialfluss, Informationsfluss, Montageprozesse, Instandhaltungsprozesse, Bewegungsabläufe
Wissensbasierte Methoden	Effiziente Möglichkeiten der Wissensrepräsentation und der Wissensverarbeitung für komplexe Problemzusammenhänge	Expertensysteme für Planung, Diagnose, wissensbasierte Modellierung von Produktionssystemen

Viele Autoren (Jain et al., 2001), (Schunk und Plott, 2000), (Ingalls, 1998), (Arnold et al., 2008)) vertreten die Auffassung, dass wegen der Variabilität, Vernetztheit und Komplexität einer Supply Chain die Simulation den besten Analyseansatz bietet. Van Der Zee and Van Der Vorst (Van Der Zee und Van Der Vorst, 2005) kommen zum Schluss, dass “In many cases, discrete

event simulation is a natural approach in studying supply chains as their complexity obstructs analytic evaluation". Huang et al. (Huang et al., 2003) bieten eine Literaturübersicht und finden ebenfalls, dass analytische Methoden hauptsächlich zur Lösung von Modellen mit geringen struktureller Komplexität verwendet werden. Schließlich führt Banks et al. (Banks et al., 2002) einige Anwendungsfälle der Simulation im Supply Chain Management auf und konstatiert, dass das Anwendungsspektrum noch nicht ausgeschöpft sei.

2.5.2 Vorteile der Simulation gegenüber anderen Analyseverfahren

- Kann zur Untersuchung von Systemen eingesetzt werden, die so komplex sind, dass sie nicht mehr praktikabel durch mathematische Modelle beschrieben werden können (Košturiak und Gregor, 1995), (Arnold et al., 2008).
- Simulation ist im Gegensatz zu mathematisch, analytischen Modellen speziell in Kombination mit graphischen Modellen relativ verständlich und damit anwenderfreundlich (Kaczmarek, 2002).
- Variationen der Eingangsparameter oder des Modells selbst können ohne Auswirkung auf das Realsystem einfach durchgeführt und analysiert werden.

Die Simulation wertet den Zustand des Modells für jeden Zeitpunkt aus, damit kann die zeitliche Entwicklung des Systemzustands als Ergebnis bereitgestellt werden und direkt im Modell visualisiert werden.

- Sofern keine real-time-Simulation angewendet wird, kann der Zeitverlauf gegenüber dem Realsystem gestaucht oder ausgedehnt werden, um die bessere Beobachtung von Phänomenen zu ermöglichen (Banks, 1999). Dies ist besonders interessant in Kombination mit der Animation.
- Simulation macht dynamische Zusammenhänge transparent und unterstützt bei der Beantwortung der Frage nach dem „Warum?" bestimmte Effekte eintreten (Banks, 1999).

2.5.3 Nachteile der Simulation gegenüber anderen Analyseverfahren

- Die Erstellung von Modellen, Befüllung mit Daten und Auswertung der Ergebnisse benötigt spezielle Kenntnisse. Für Banks (Banks, 1999) ist die Beherrschung von Simulationsverfahren „an art that is learned over time and through experience".
- Simulation liefert keine optimale Lösung, sie hilft die Zusammenhänge zu verstehen und das untersuchte Modell zu bewerten. Dies ist im Speziellen der Fall, wenn die Simulationsmodelle zu komplex werden, um effektive Optimierungsverfahren einzusetzen (Almeder und Preusser, 2007b, S. 1932). Es wird immer eine Heuristik bei

der Anwendung der Simulation benötigt, um zu verbesserten Ergebnissen zu gelangen.
- Die Modellerstellung, Sammlung von Daten und die Durchführung selbst kann lange dauern, teuer und rechenintensiv sein.

Ergänzend zu den aufgelisteten Nachteilen präsentieren Banks und Gibson (Banks und Gibson, 1997) in ihrem Paper 10 Regeln, wann Simulation zur Auswertung spezifischer Anwendungsfälle nicht das richtige Instrument ist und in (Banks und Chwif, 2011) werden von Banks und Chwif Warnungen bezüglich häufiger Fehler in der Simulationsanwendung besprochen.

Abschließend kann entschieden werden, dass der Simulationsansatz das richtige Instrument ist, um die mit der Forschungsfrage verbundenen Analyseanforderungen erfüllen zu können.

2.5.4 Klassifikation von Simulationsverfahren

Simulationsverfahren werden in der Literatur auf verschiedene Art und Weise klassifiziert. Um zu ermitteln welches Simulationsverfahren für die, in dieser Arbeit behandelte Problemstellung geeignet ist, werden im Folgenden Charakteristika von Supply Chain Modelle untersucht und der geeignete Simulationsansatz abgeleitet:

Klassifikation nach Zeitabhängigkeit (Košturiak und Gregor, 1995, S. 3) des Modells:

- Statische Simulationsmodelle, repräsentieren ein System zu genau einem Zeitpunkt (Banks et al., 2004, S. 11), ein typisches Beispiel ist die Monte Carlo Simulation.
- Dynamische Simulationsmodelle repräsentieren ein System und seine Veränderungen im Zeitverlauf.

Da in dieser Arbeit die zeitliche Synchronisation von Informations- und Materialflüssen von elementarer Bedeutung ist, wird ein Simulationsverfahren benötigt, das die Auswertung dynamischer Modelle erlaubt.

Klassifikation nach der Zeitverteilung der Eingangs-, Ziel- und Stellgrößen des Systems. Zustandsänderungen in Prozessen können diskret oder stetig erfolgen (Košturiak und Gregor, 1995, S. 4).

Diskretes Simulationsmodell: Die Zustandsvariablen ändern sich nur zu definierten Zeitpunkten. Ein Produktionsauftrag ist gestartet oder eine Anlage ausgefallen. Auch wenn in Bezug auf Informations- und Materialfluss von einem „Fluss“ die Rede ist, ist für das Fortschreiten der Supply Chain nur relevant, ob das Teil oder die Information zum richtigen Zeitpunkt an der Bearbeitungsstation oder dem Supply Chain Partner vorhanden ist.

Stetiges (oder kontinuierliches) Simulationsmodell: Die Zustandsvariablen ändern sich kontinuierlich im Zeitverlauf. Beispielsweise kann es bei chemischen Prozessen zu stetigen, kontinuierlichen Zustandsänderungen kommen. In Bezug auf die Domäne Instandhaltung ist anzumerken, dass, obwohl sich der Zustand eines instand zu haltenden Guts über die Zeit stetig verschlechtern kann, handelt es sich trotzdem um einen diskreten Prozess, da nur eine zeitpunktbezogene Inspektion[41] die Zustandsveränderung misst und somit wahrnimmt.

Viele Modelle enthalten sowohl stetige auch als diskrete Zustandsänderungen, zur Klassifikation wird dann das dominante Verhalten herangezogen (Law und Kelton, 1991). Supply Chain Netzwerke sind daher als diskrete Modelle anzusehen.

Klassifikation nach dem Grad der Bestimmtheit:

Deterministische Systeme: Das deterministische Verhalten ihrer Eingangs- und Stellgrößen ermöglicht eine exakte Vorhersage des Verhaltens des Systems.

Stochastische Systeme: Hat ein oder mehrere Zufallsvariablen als Eingangsgrößen (Banks et al., 2004). Durch das stochastische Verhalten ihrer Eingangs- und Stellgrößen variiert das Verhaltens des Systems und damit die Zielgrößen[42] zufallsbedingt, eine exakte Vorhersage ist nicht möglich. Im Bereich der Supply Chain wird daher mit einer großen Stichprobe und Ermittlung von Erwartungswerten gearbeitet.

Wobei eine Unterscheidung in Stochastische Simulation und Deterministische Simulation laut (Košturiak und Gregor, 1995) nicht sinnvoll ist, da in der Praxis selten alle Variablen eines Modells rein deterministisch oder stochastisch sind und Zufallsgrößen mithilfe von deterministischen Verfahren umgesetzt werden. Da aber in vielen Fragestellungen die mittels Simulation für Maintenance Supply Chains beantwortet werden sollen, zumindest eine Eingangsgröße stochastischer Natur ist, muss die Simulation dies unterstützen.

Zusammengefasst kann gesagt werden, dass Modelle die mit dem, in dieser Arbeit entwickelten Ansatz erstellt werden, dynamisch, diskret und stochastisch sind. Zur Analyse von Modellen mit diesen Charakteristika ist die zeitdiskrete Simulation das am besten geeignete Analyseinstrument. Diese Schlussfolgerung deckt sich auch mit dem Großteil der Literatur zu diesem Thema (Robinson, 2014, S. xviii).

[41] Auch ein online-Zustandsmonitoring, obwohl beispielsweise als Teil eines SCADA Systems misst Indikatoren für den Zustand der Anlage (Geräusch, Vibration, Strombedarf..) nur in Intervallen, wenn auch in vergleichsweise kurzen Zeitabständen

[42] Sofern ein direkter Zusammenhang zwischen stochastischer Eingangs- und Zielgröße besteht

2.5.5 Diskrete Simulation

Wie oft und wann das Simulationsmodell neu berechnet wird, um eine Zustandsänderung zu ermitteln, ist ein weiteres Unterscheidungsmerkmal von Simulationsalgorithmen. Die Simulationsuhr, welche die, in einem Simulationslauf bisher vergangene Zeit anzeigt, wird dabei entweder in konstanten Schritten vorgerückt oder aber auf den chronologisch nächsten Ereigniszeitpunkten gestellt.

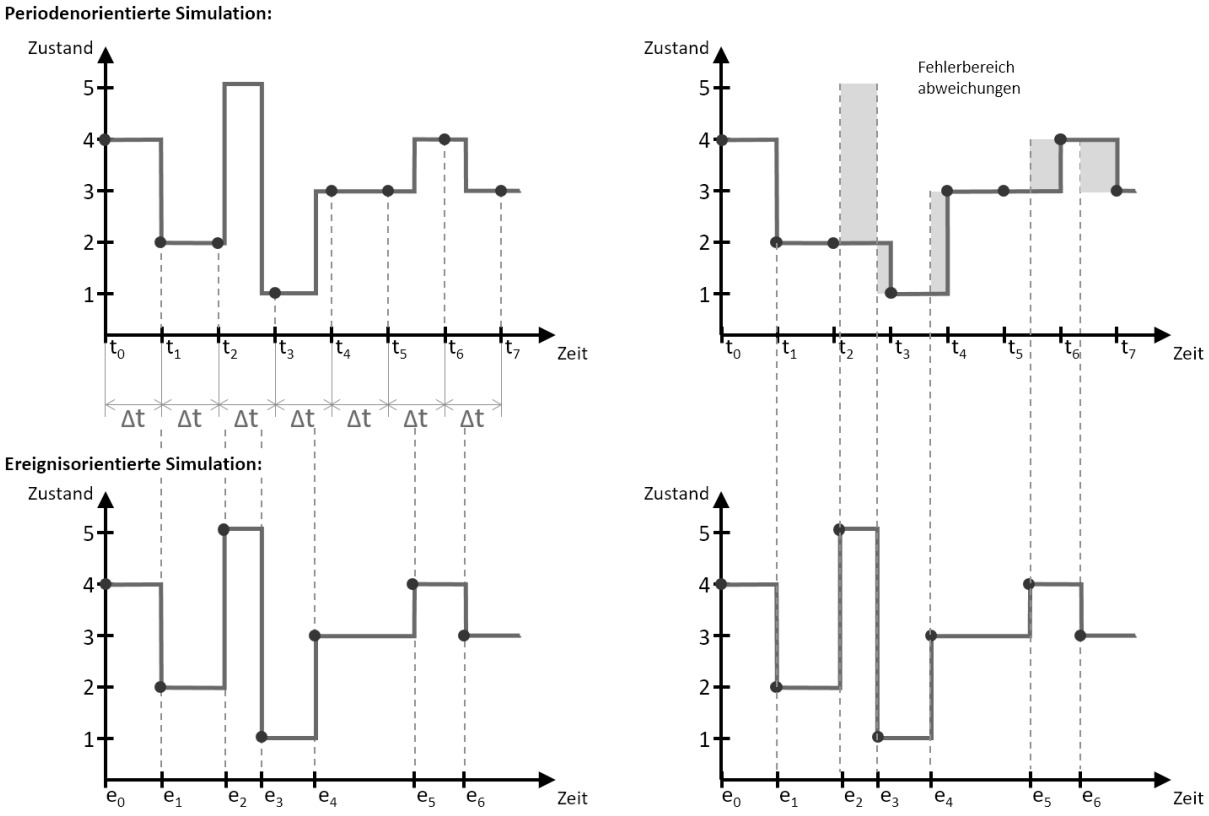


Abbildung 29: Vergleich periodenorientierter mit ereignisorientierter Simulation, Teile der Grafik übernommen aus (Hedtstück, 2013)

2.5.5.1 Periodenorientierte Simulation

Der Simulationsalgorithmus prüft und aktualisiert die Systemgrößen in periodischen, gleichbleibenden Zeitabständen (Hedtstück, 2013, S. 30). Diese Art der Simulation wird auch als zeitorientierte, zeitgesteuerte, synchrone oder Zeitscheibensimulation bezeichnet.

Nachdem der Systemzustand am Ende der Periode i , also zum Zeitpunkt t_i ermittelt wurde, wird die Simulationsuhr um eine Periodendauer Δt vorgerückt, es gilt also

$$t_{i+1} = t_i + \Delta t$$

Dabei muss es sich bei einem Zeitpunkt t_i nicht um einen konkreten Zeitpunkt (wie zum Beispiel 14:22 Uhr) oder ein Datum handeln, oft werden einfach numerische Periodenzähler

verwendet. Die Verwendung von Periodenzähler erlaubt eine gewisse Flexibilität, da die Periodendauer frei gewählt werden kann. So kann beispielsweise die Periodendauer ein Monat sein, um das dynamische Verhalten über einen langen Zeitraum untersuchen zu können, wie es für strategische Fragestellungen interessant sein kann. Soll hingegen das taktische Verhalten analysiert werden, kann ein Tag die richtige Periodendauer sein, wogegen für operationelle Entscheidungen Zustandsänderungen im Stunden oder Minutenbereich auswertbar sein sollten.

Die Ablaufsteuerung einer periodenorientierten Simulation kann folgendermaßen implementiert werden (Hedtstück, 2013):

```
001 tSimulationsEnde:( USER_ENTRY )
002 WHILE (tSimulationsZeit < tSimulationsEnde)
003 {
004   tSimulationsZeit:( tSimulationsZeit + tPeriodenDauer );
005   DO „berechne den Systemzustand neu“;
006 }
```

Wobei zur Neuberechnung des Systemzustandes eine Liste aller Bearbeitungsstationen durchlaufen werden muss.

Vorteile dieser Methode sind:

- einfach zu implementieren
- die Kompression der realen Zeit erfolgt während des gesamten Simulationslaufs im selben Verhältnis. Dies ist ein Vorteil für eine Echtzeit-Animation während der Simulation, da es durch die Uniformität des Zeitfortschritts zu keinen Zeitsprüngen kommt

Nachteile dieser Methode sind:

- Ereignisse, die während einer Periode auftreten werden erst am Ende der Periode ausgewertet. Dies führt zu Ungenauigkeiten, die durch Verkürzung der Periodendauer verringert werden können, siehe Abbildung 29
- rechenintensiv, da der Modellzustand auch neu berechnet werden muss, wenn keine Veränderung stattfand
- Wahl der richtigen Periodendauer ist wichtig, aber nicht immer einfach

2.5.5.2 Ereignisgesteuerte Simulation

Die diskrete ereignisgesteuerte Simulation betrachtet nur die Zeitpunkte, in denen tatsächlich Änderungen im Systemzustand stattfinden. Diese Momente werden als Ereignisse bezeichnet. Ein Ereignis in sich selbst verbraucht keine Zeit und ist atomar, es kann also nicht von einem anderen Ereignis unterbrochen werden. Das Verhalten von Objekten des Systems erzeugt

Ereignisse in der Zukunft, welche in eine chronologische Ereignisliste aufgenommen werden. Der Treiber der Dynamik ist also nicht die Simulationsuhr, sondern ein Algorithmus, der die einzelnen Ereignisse der Ereignisliste nacheinander abarbeitet und ihren Effekt auf das System ermittelt. Ereignisse können zum selben Zeitpunkt eintreten, werden aber durch den Simulationsalgorithmus serialisiert, also in einer bestimmten Reihenfolge abgearbeitet.

Die Ablaufsteuerung einer ereignisgesteuerten Simulation kann folgendermaßen implementiert werden (Hedtstück, 2013):

```
001 WHILE ( tSimulationsZeit < tSimulationsEnde )
002 {
003   „Hole das nächste Ereignis eNächstesEreignis aus der Ereignisliste
      und lösche diesen Eintrag“;
004   tSimulationsZeit:( „Eintrittszeitpunkt von eNächstesEreignis“ );
005   sEreignisTyp = „Typ von nächstesEreignis“;
006   „Führe die Ereignisroutine zu ereignisTyp aus“;
007 }
```

Vorteile dieser Methode sind:

- effizient, da nur die Zustandsänderungen von Objekten, die vom Ereignis direkt beeinflusst werden, ermittelt werden müssen
- genauere Simulationsergebnisse im Vergleich zur periodenorientierten Simulation

Nachteile dieser Methode sind:

- aufwendiger zu implementieren als die periodenbezogene Simulation

Bezüglich der diskreten Simulation werden drei verschiedene Vorgehensweisen unterschieden[43] (Schwetman, 1986):

- Ereignisorientierte Simulation
- Aktivitätsorientierte Simulation
- Prozessorientierte Simulation

In einer reinen *Ereignisorientierten Simulation* werden Aktivitäten nicht direkt betrachtet, sondern nur die, zu Ereigniszeitpunkten durch die Aktivitäten ausgelösten Zustandsänderungen verfolgt. Dies erlaubt eine Trennung zwischen der statischen Struktur des Modells und seinem dynamischen Verhalten. Die Ereignisse werden auf einer Zeitachse aufgereiht und der Reihe nach bearbeitet. Da nur die tatsächlichen Veränderungen von der

[43] Page (Page, 2013), der die Implementierung von Simulationsalgorithmen zum Inhalt seines Buches macht, fügt noch eine vierte Sicht die „Transaktionsorientierte Sicht“ hinzu.

Simulation betrachtet werden, ist diese Vorgehensweise sehr effizient, kann aber dadurch, dass viele Ereignisse an unterschiedlichen Stellen im Modell auftreten können, schnell unübersichtlich werden.

Die *Aktivitätsorientierte Simulation* (Page, 2013, S. 29) basiert darauf, dass alle Bearbeitungsstationen, in denen das Material einer Eigenschaftsveränderung unterzogen wird, über Start- und Endbedingungen verfügen, die erfüllt sein müssen, um mit der Bearbeitung zu beginnen oder sie zu beenden. Dabei können diese Bedingungen beliebig komplex ausgestaltet werden, wie zum Beispiel, dass die benötigten Materialien in der richtigen Menge vorliegen, die Bearbeitungsstation nicht belegt ist, der Bediener der Maschine verfügbar ist und ein Auftrag vorliegt. Das Problem dabei ist, dass alle diese Bedingungen in bestimmten Zeitabständen ausgewertet werden müssen und dies zum einen ineffektiv sein kann und zum anderen, wenn die Zeitabstände schlecht gewählt wurden, zu ungenauen Simulationsergebnissen führen kann.

Die *Prozessorientierte Simulation* fügt alle Aktivitäten, die ein dynamisches Objekt durchläuft, zu einem Prozess zusammen. Die Bearbeitungsstation selbst wird auch durch einen Prozess abgebildet. Prozesse können nach der Bearbeitung in einen inaktiven Zustand versetzt werden, beziehungsweise später wieder reaktiviert werden. Technisch gesehen, wird die Prozessorientierte Simulation auf eine ereignisorientierte Simulation aufgesetzt. Die Programmierung der Ablaufkontrolle zur Koordination von parallelen Prozessen kann sehr aufwändig werden und erfordert die Verwendung einer eigenen Programmiersprache (Schwetman, 1986).

Diskrete ereignisgesteuerte Simulation kann auch aus der Sicht der Ergebnisdaten klassifiziert werden (Banks et al., 2002, S. 285):

Eingeschwungene Simulation „steady-state“: Sie wird üblicherweise verwendet um das relative Langzeitverhalten eines Systems zu untersuchen, beispielsweise um die durchschnittlichen Durchlauzeiten durch die Supply Chain zu ermitteln.

Terminierende Simulation: Sie wird verwendet um beispielsweise die Zeit zu bestimmen, bis die Supply Chain eine bestimmte Anzahl von Gütern liefern kann. Die Simulation startet in einem ruhenden System, das heißt dass Bearbeitungsstationen nicht belegt sind und keine Warteschlangen vorhanden sind und endet mit einem definierten Endereignis.

2.6 Animation

„Logicians may reason about abstractions.
But the great mass of men must have images.“

Thomas Babington Macaulay

Laut Rohrer (Rohrer, 2000), der auch mit obigem Zitat sein Paper einleitet, ist die Visualisierung der Dynamik eine der bedeutendsten Komponenten von Simulationsprogrammen geworden. Schon die Entwicklung graphischer Modellerstellung basiert auf der Annahme, dass Bilder von Menschen besser verstanden und schneller verarbeitet werden können als inhaltlich gleichwertige mathematische Ausdrücke oder textuelle Beschreibungen. Eine der fundamentalen Hypothesen dieser Arbeit ist, dass die Animation von Flüssen, also die Bewegung von dynamischen Objekten durch die Supply Chain, es vielen Fachexperten erleichtert, Zusammenhänge im Netzwerk zu erkennen und Ideen zur Verbesserung zu entwickeln (Tversky et al., 2002), (Park und Gittelman, 1992), (Thompson und Riding, 1990).

Animation von Simulationsmodellen kann in real-time, während der Simulationsberechnung, oder aber zu einem späteren Zeitpunkt, als Play-back, erfolgen. Nachteil der real-time Animation ist, dass insbesondere wenn die Simulation sehr lange dauert, der Beobachter an Konzentration verliert und eventuell wichtige Systemzustandsänderungen nicht erfasst. Außerdem können Ausschnitte des Simulationslaufs nicht wiederholt werden, was bei stochastischen Einflüssen zu besonderen Problemen führt, um sie beispielsweise einem breiteren Publikum zu demonstrieren. Viele Simulationstools bieten deshalb die Funktionalität an, die Animation sozusagen aufzuzeichnen, um sie später beliebig oft wiederholen zu können. Carson (Carson, 2002, S. 57) empfiehlt die Animation überhaupt nur zur Untersuchung kurzer Zeiträume heranzuziehen.

Vorteile der Animation sind (Rohrer, 2000):

- erleichtert Verifikation und Validierung von Modellen und Simulationsergebnissen
- unterstützt das bessere und schnellere Verständnis der dynamischen Zusammenhänge
- hilft bei der Kommunikation der Simulationsergebnisse
- demonstriert die Realitätsnähe des Modells und erhöht damit die Glaubwürdigkeit des Modells

Ein Nachteil kann sein, dass ein übermäßiges Vertrauen in die Simulation gesetzt wird, ohne die Beschränkungen, die der Modellerstellung unterlegen sind, zu berücksichtigen.

Rabe et al. (Rabe et al., 2008, S. 95f) untersuchen die Eignung der Animation zur Verifikation und Validierung von Simulationsmodellen. Laut Law (Law, 2008, S. 45) kann Animation zeigen, dass für bestimmte Situationen und Zeiträume das Verhalten eines Modells ungültig

ist. Ein Umkehrschluss ist allerdings nicht möglich, da selten auftretende Fehler im Modell mit hoher Wahrscheinlichkeit während der Betrachtung der Animation nicht auftreten oder nicht auffallen[44].

2.7 Abbildung von Dynamischem Modellverhalten

In Anlehnung an die Definition[45] von Sommer (Sommer, 2011, S. 96) wird in dieser Arbeit unter dem dynamischen Verhalten eines Systems seine Aktionen oder Reaktionen auf eintretende, externe oder interne Ereignisse im Zeitverlauf verstanden. Eine Aktion kann dabei beispielsweise eine Interaktion mit seiner Umwelt oder die Veränderung einer Eigenschaft des Systems selbst sein. Um modellhaft die dynamische Entwicklung eines Systems zu untersuchen, muss das statische System und sein dynamisches Verhalten in einem Modell abgebildet werden. Dieses Kapitel beschäftigt sich mit den Möglichkeiten der Modellierung des dynamischen Verhaltens.

Systeme, deren Komponenten zueinander in einer unveränderbaren Beziehung stehen sind *statische Systeme* (Schlichting, 2004, S. 17). Verändern sich die Beziehungen zwischen den Objekten jedoch, im Falle der Supply Chain durch Transport von Materialien oder Kommunikation mittels dem Informationsfluss, so spricht man von einem dynamischen Modell (Schlichting, 2004, S. 17). Supply Chains, die sich stetig an sich ändernde Umwelteinflüsse anpassen müssen, sind demnach dynamische Systeme.

Das Verhalten eines Systems wird beeinflusst vom Verhalten seiner Systemkomponenten. Bildet man das System in einem Modell ab, so wird das Modellverhalten von den einzelnen Modellkomponenten oder Modellierungsobjekten bestimmt. Das Verhalten einer Supply Chain wird also von den einzelnen Supply Chain Partnern beeinflusst und dieses wiederrum vom Verhalten ihrer Ressourcen, welche die einzelnen Planungs- oder Bearbeitungsschritte durchführen. Spricht man auch den einzelnen dynamischen Objekten, die durch den Materialfluss transportiert werden, ein Verhalten zu, was beim Einsatz von RFID-Technologie denkbar ist, so kann jedes in der Supply Chain bearbeitete Produkt das Verhalten des Gesamtmodells beeinflussen. Kennt man das Verhalten der Einzelteile eines Modells, bedeutet das aber nicht zwangsläufig, dass man das Verhalten des Gesamtmodells einfach ableiten kann. Das Gesamtsystem kann Eigenschaften aufweisen, die bei Betrachtung kleiner Ausschnitte gar nicht sichtbar oder erkennbar werden (Jakoby, 2013, S. 16). Speziell bei dynamischen Systemen, bei denen die Ausgangsgrößen, auch von zeitlich weiter zurückliegenden oder gar rückgekoppelten Eingangsgrößen abhängen, wird eine Vorhersage

[44] Ehrig (Ehrig, 2003) führt diese Thematik noch weiter aus

[45] Er definiert hier „Past Behaviour“

des Systemverhaltens schwierig und Bedarf einer dynamischen Analysemethode, der Simulation.

Davis (Davis, 1988) hat in seinem Aufsatz verschiedene Techniken zur Spezifikation des externem Systemverhalten von Software verglichen, darunter Finite State Machines, Entscheidungstabellen und Entscheidungsbäume, Softwarecode / Programme, Statecharts und Petri Netze. Im Folgenden wird auf graphische und verbale Ansätze zur modellhaften Beschreibung des dynamischen Verhaltens eines Systems eingegangen.

2.7.1 Graphische Modelle zur Definition von Verhalten

Wie im Kapitel 2.3 ausgeführt, sind ein Charakteristikum einer Maintenance Supply Chain die komplexen und hochdynamischen Wechselwirkungen zwischen den einzelnen SC-Partnern und deren individuellen Steuerungs- und Ausführungsprozessen. Das Verhalten eines Objekts in der Supply Chain ist eine Aktion oder Reaktion auf objektexterne Einflüsse, die über den eintreffenden Material- oder Informationsfluss, die Zeit oder andere globale Ereignisse auf das Objekt einwirken können. Die Reaktion darauf kann über den austretenden Material- oder Informationsfluss oder der Veränderung einer Objekteigenschaft, wie zum Beispiel dem Status, der Bearbeitungszeit oder des Ressourcenverbrauchs eines Produktionsschritts erfolgen.

2.7.1.1 Verhaltensmodellierung mittels Geschäftsprozessen

In Anlehnung an Stott und Crosslin (Stott und Crosslin, 1992, S. 38) verstehen Kruse und Scheer (Kruse und Scheer, 1992) unter „Dynamischer Modellierung“ die Beschreibung des Verhaltens von betriebswirtschaftlichen und informationstechnischen Systemen und deren Simulation. Dabei verweisen sie für die Beschreibung des Systemverhaltens auf die Geschäftsprozessmodellierung, mit der Aspekte wie Zeitbedarf, Quantitäten und logisch-kausale Zusammenhänge, wie beispielsweise Systemzustände, Ereignisse oder Ressourcen abgebildet werden können.

Beschränkt man die Beschreibung des Verhaltens eines Objekts auf Aktivitäten, die nach dem Eintreten eines Ereignisses einem Kontrollfluss folgend abgearbeitet werden, so kann jede kontrollflussorientierte Geschäftsprozess-Modellierungssprache, wie beispielsweise die entsprechenden Modelltypen der BPMN, ADONIS BPMS, EPK oder UML zur Definition des Verhaltens verwendet werden.

UML (Fowler, 2004), eine Modellierungssprache, die zur Spezifikation und Dokumentation von Anwendungssoftware entwickelt wurde, bietet neben Diagrammtypen zur Beschreibung der statischen, strukturellen Komponenten eines Systems auch Diagrammtypen zur Abbildung von Systemverhalten an. In der aktuellen Version UML 2.5 (OMG, 2015) sind das

wie aus Abbildung 30 ersichtlich, die kontrollflussorientierten Activity Diagrams, die State Machine Diagrams und die Interaction Diagrams. Auch SysML (Weilkiens, 2006), eine auf UML 2.0 aufbauende Modellierungssprache, die als Standard für den Bereich des Systems Engineering, also dem Entwurf elektrischer, mechanischer, chemischer und softwaretechnischer Systeme entwickelt wurde und somit für den Produktionsbereich interessant ist, verfügt über Modelltypen mit denen Verhalten beschrieben werden kann. Schönherr und Rose zeigen in ihrem Aufsatz (Schönherr und Rose, 2010) eine entsprechende Anwendung.

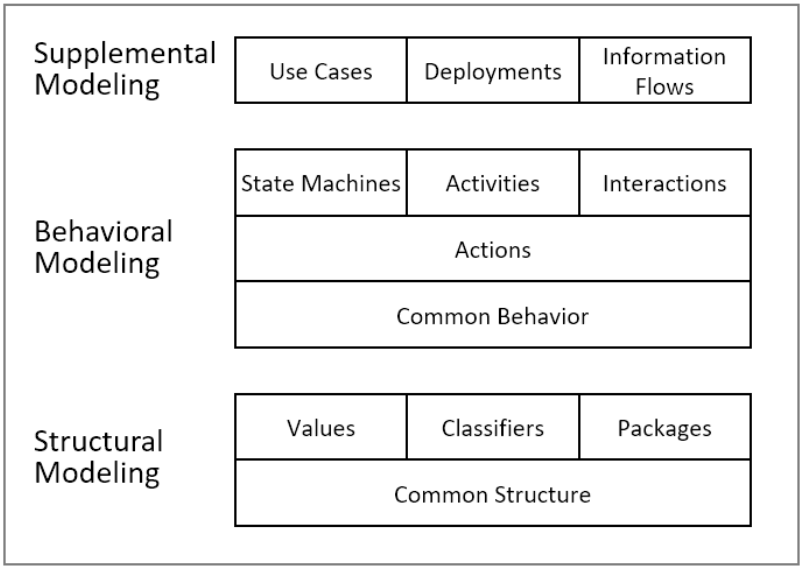


Abbildung 30: Semantische Bereiche der UML (OMG, 2015)

2.7.1.2 Verhaltensmodellierung mittels graphischer Entscheidungsbäume

Werden die zuvor beschriebenen Reaktionen oder fachlichen Entscheidungsprozesse in einer dafür geeigneten kontrollflussorientierten, graphischen Modellierungssprache abgebildet, so muss für jede Entscheidungssituation ein Entscheidungsbaum modelliert werden. Da die gängigen Modellierungssprachen keinen Modelltyp „Entscheidungsbaum“ kennen, wird naheliegender Weise der Entscheidungsprozess modelliert.

Abbildung 31 zeigt beispielhaft wie eine Verhaltensdefinition mittels Ereignisgesteuerten Prozessketten aussehen kann. Das Szenario zeigt das Verhalten des Prozesses „Materialeingang bearbeiten“ im Kontext des prozessexternen Material- und Informationsflusses. Das Eintreffen eines Materials vom Lieferanten (über den Materialfluss) löst das Ereignis „Material eingetroffen“ aus. Eine positive Qualitätskontrolle führt zum Weiterleiten des Materials über den Materialflusskanal an das Zwischenlager und Versand einer Information über den Informationsflusskanal an die Produktionsplanung. Ist die Qualität jedoch mangelhaft, so wird das Material über den Materialflusskanal an den Lieferant zurückgesendet und der Lieferant darüber informiert.

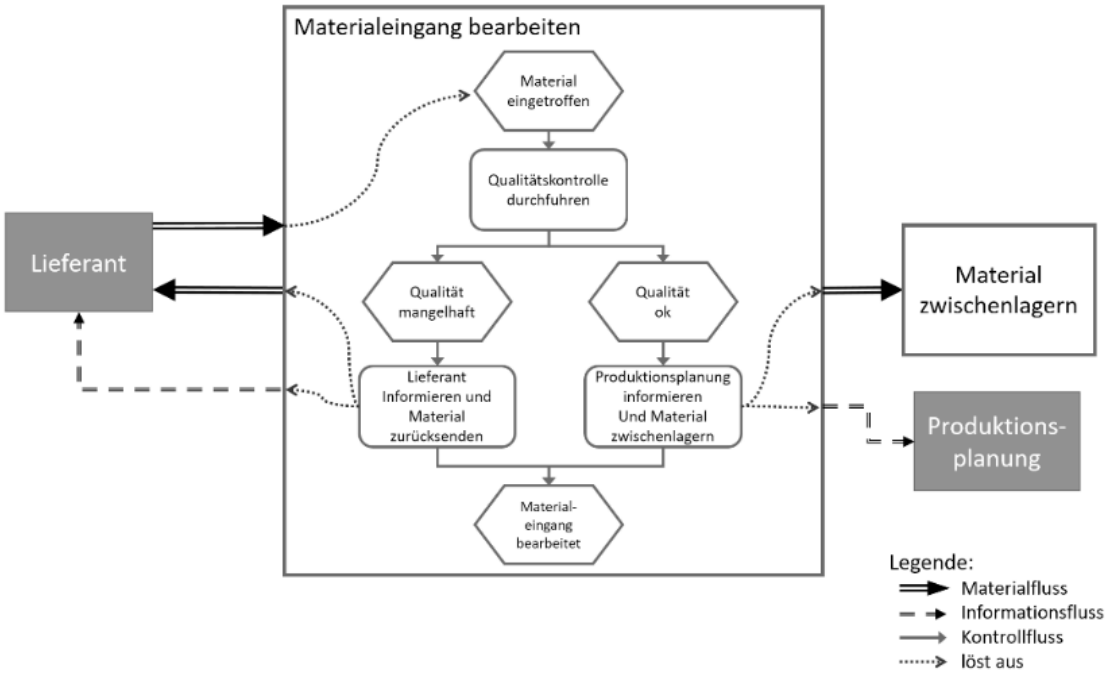


Abbildung 31: Geschäftsprozess zur Beschreibung von material- und informationsflussspezifisches Verhalten

Dieses Beispiel zeigt, dass einfache Verhaltenssteuerung eines Objekts mittels einer kontrollflussorientierten Modellierungssprache möglich ist.

Die in dieser Arbeit betrachteten Supply Chains bestehen jedoch aus einem komplexen Geflecht von untereinander, mittels Material- und Informationsflüssen interagierenden Objekten. Darüber hinaus ist das Objektverhalten oft abhängig von einer Kombination verschiedener Eingangsgrößen, wie beispielsweise Bestandsgrößen, Ressourcenverfügbarkeit und Nachfrage. Die sich daraus ergebende Vielzahl von Entscheidungsalternativen macht deren Ausmodellierung in einem Geschäftsprozessmodell nicht mehr effizient.

Abbildung 32 zeigt als Beispiel einen Extrakt aus einem einfachen Prozess, der den Lagerstand von Lager A abfragt und wenn dieser kleiner als 5 ist, den Lagerstand von Lager B abfragt. Ist dieser kleiner als 3 so wird eine Bestellung ausgelöst.

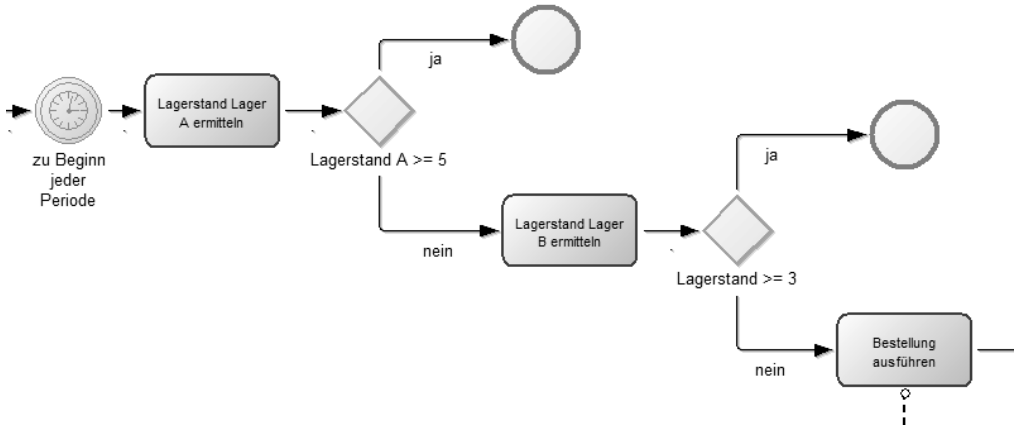


Abbildung 32: Beispiel eines Entscheidungsbaums modelliert in BPMN

Dieses einfache Beispiel mag noch überschaubar und plausibel sein, ergänzt man das Modell aber noch um eine weitere Einflussgröße, wie zum Beispiel den Standort des Lagers oder ein nachfragebezogenes Ereignis, bei dessen Eintreten die Lagergrenzen (im Beispiel 5 und 3) verändert werden, so steigt in der Maximalausprägung die Anzahl der Entscheidungsknoten und damit die Anzahl möglicher Alternativpfade pro Einflussgrößen um den Faktor 2 an. Für jeden Produktionsvorgang oder für jede logistische Aktion im Netzwerk muss ein eigener Entscheidungsprozess grafisch modelliert werden. Der Einschätzung und Erfahrung des Autors nach, leidet darunter die Lesbarkeit des Modells und damit die Verständlichkeit der Analyseergebnisse sehr. Gerade bei einer größeren Menge von beteiligten Objekten innerhalb eines sich ständig ändernden Systems ist die Menge der möglichen Ausprägungen, der Interaktionen und daraus folgenden Reaktionen so groß, dass eine graphische Modellierung aller Alternativkombinationen nicht realistisch abbildbar ist.

Viele der Aktionen und Reaktionen, die das Verhalten einzelner Supply Chain-Objekte prägen, werden durch die Fachlichkeit oder Entscheidungen aus dem Unternehmen vorgegeben. Es liegt in der Natur der Sache, dass in einer Maintenance Supply Chain, die vielen, auch extern verursachten Einflüssen unterliegt, sich diese Vorgaben ändern beziehungsweise schon im vornhinein flexibel gestaltet werden müssen. Die Änderung eines einzelnen Werts in einem graphischen Prozess, der eine Entscheidungssituation abbildet, ist erfahrungsgemäß schnell umgesetzt. Ändert sich aber die Struktur der Entscheidungsfindung, also der Prozess an sich, oder müssen weitere Einflussgrößen im Modell berücksichtigt werden, so hat dies häufig eine aufwändigere graphische Anpassung des Prozesses zur Folge. Dies geht zu Lasten einer effektiven Anwendbarkeit der Methode, gerade bei vergleichenden Analysen von Umsetzungsalternativen.

2.7.1.3 Berechnung von Eingangsgrößen

In typischen Produktions-, Lager und Transportprozessen, welche eine Supply Chain bilden, wird das Verhalten der einzelnen Akteure häufig von numerischen Stellgrößen gesteuert.

Beispiele dafür sind, der Start der Produktion, die Anzahl der zu transportierenden Güter oder die verstrichene Lagerzeit. Um zu diesen verhaltensbeeinflussenden Entscheidungsgrößen zu gelangen, müssen Inputs aus verschiedenen Quellen bezogen und einer Kalkulation unterzogen werden. Ein gutes Beispiel dafür ist die Steuerung von Produktionsprozessen, deren Produktionszahlen aus einer Prognoserechnung basierend auf aktuellen Verkaufszahlen abgeleitet wird.

Dieses Beispiel zeigt, dass für die Beschreibung des dynamischen Verhaltens einer Supply Chain eine Rechenfähigkeit der Beschreibungssprache notwendig ist. Eine graphische Abbildung dieser Berechnungsschritte ist in einem Modell ist möglich, wie dies beispielsweise die Umsetzung der Modellierungssprache des Balanced Score Card Tools ADOscore[46] von BOC in Abbildung 33 zeigt. Ein weiteres Beispiel aus einer anderen Softwarekategorie ist der graphische XPresso-Editor von Cinema4D®[47], einem Grafikprogramm zur Modellierung, Animation und Rendering von 3D-Modellen.

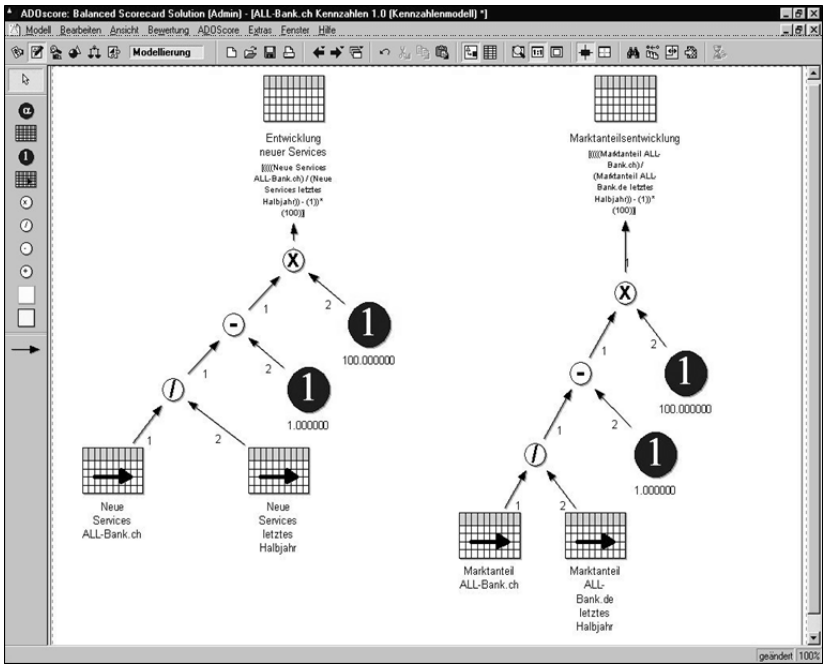


Abbildung 33: Beispiel für graphische Darstellung von Formeln in ADOscore[46]

Dennoch bietet der vorhandene Symbolvorrat nicht die Flexibilität einer Formelsprache. Außerdem entspricht der Meinung des Autors nach, eine graphische Ausmodellierung von Berechnungsformeln nicht dem Sprachgebrauch eines typischen Anwenders der Methode SIMchronization. Dies sind Fachspezialisten aus der Instandhaltung, Business Analysten und IT-Spezialisten die aus Tabellenverarbeitungsprogrammen gewohnt sind, Berechnungen als textuelle Formeln anzugeben.

[46] ADOscore ist ein eingetragenes Markenzeichen der BOC AG

[47] Cinema4D® ist ein eingtragenes Markenzeichen der MAXON GmbH; Informationen zu XPresso sind unter der Website https://help.maxon.net/de/#5828 (Zugriff am 07.01.2017) zu finden.

2.7.1.4 Verbale Beschreibung von dynamischem Objektverhalten

Eine Alternative zur graphischen Modellierung ist eine textuelle Beschreibung des Verhaltens der Objekte. Diese kann frei oder formalisiert erfolgen. Eine freie textuelle Formulierung hat in der Erhebungs- und Diskussionsphase mit dem Fachbereich den Vorteil, dass die Formalismen nicht näher erläutert werden müssen, da die Syntax als bekannt vorausgesetzt werden kann und somit mehr Kapazität für die Diskussion der fachlichen Sachverhalte bleibt[48]. Der große Nachteil ist jedoch, dass ein freier Text nicht ohne weiteres maschinell ausgewertet werden kann, eine softwaregestützte Dynamisierung des Modells ist damit nicht möglich. Deshalb wird die freie Beschreibung nur initial in der Erhebungsphase und später in der Test- und Dokumentationsphase verwendet.

Um zu einem, vom Simulationsalgorithmus auswertbaren Modell, bestehend aus statischen und dynamischen Bestandteilen, zu kommen, muss diese freie verbale Verhaltensbeschreibung in ein formales, einer bekannten Struktur folgendes, Konstrukt übersetzt werden. Diese Tätigkeit wird üblicherweise von einem Business Analysten durchgeführt.

2.7.2 Business Rules

Jede Regel reduziert Freiheitsgrade in der betroffenen Domäne (OMG, 2008), indem sie verbindlich geltende Richtlinien vorgibt. Für die Business Rules Group[49] ist eine Business Rule (Hay und Healy, 2000):

"… a statement that defines and constraints some business. It is intended to assert business structure or to control or influence the behavior of the business"

Die Object Management Group, Inc. OMG (OMG, 2009) versteht unter Business Rules die Regeln, für deren Ausgestaltung diejenigen, die das Geschäft verantworten, zuständig sind.

"a rule that is under business jurisdiction, which means that the business can enact, revise, and discontinue business rules as it sees fit."

Unter einer Business Rule oder Geschäftsregel kann also eine von Seiten des Unternehmens vorgegebene Richtlinie verstanden werden. Der Beiname „Business" suggeriert zwar eine direkten Bezug der Richtlinie zur Geschäftstätigkeit mit Kundenkontakt oder Compliance-Vorgaben für Mitarbeiter, Business Rules können aber auch dafür eingesetzt werden, Vorgaben (Axiome) aus Sicht der Informationstechnologie zu formulieren. Die Business Rules

[48] siehe auch Kapitel 9 „Modellierungsverfahren"

[49] vormals auch unter dem Namen „GUIDE Business Rules Project" bekannt

Group fokussiert sich speziell auf diesen Aspekt und spezialisiert ihre oben angeführte Definition einer Business Rule zu (Hay und Healy, 2000).

„a business rule expresses specific constraints on the creation, updating, and removal of persistent data in an information system"

In jeder Organisation existiert eine Vielzahl verschiedener Business Rules, die von nicht dokumentierten Gepflogenheiten im Umgang mit Kunden, zu dokumentierten Arbeitsanweisungen, bis zu geheimen, im IT-System integrierten Prüfregeln für Vorgänge reichen. Taveter und Wagner (Taveter und Wagner, 2001) teilen Business Rules in folgende Kategorien ein:

Derivation rules (Ableitungsregeln), deren Anwendung neue Informationen durch Schlussfolgerungen oder mathematische Berechnungen (Rosenberg und Dustdar, 2005) aus bestehenden Fakten generiert. Ein Beispiel wäre „Ein Ersatzteiltyp ist kritisch, wenn er der Klasse A entspricht und sein Lagerstand kleiner gleich eins ist."

Integrity constraints (Einschränkungen), sind Regeln, deren Ergebnis bei Anwendung immer wahr sein muss. Dies entspricht einem Verbot oder einer Richtlinie. Ein Beispiel wäre „Eine Bearbeitungsstation, die in der aktuellen Periode produziert, darf keinen neuen Auftrag annehmen"

Reaction rules (Aktionsregeln), sind Regeln die bei Eintreffen eines Ereignisses oder der Erfüllung einer Bedingung weitere Aktionen auslösen. Ein Beispiel wäre „Wenn ein Teil im Pufferlager eintrifft, wird es sofort einer Qualitätsprüfung unterzogen". Wagner (Wagner, 1998) schlägt folgende Definition für Reaction Rules vor:

"Reaction rules define the behavior of an agent in response to environment events (perceived by the agent), and to communication events (created by communication acts of other agents)"

In (Hay und Healy, 2000) wird eine vierte Kategorie, die Autorisierungsregel („deontic assignments") vorgeschlagen. Diese Regeln können Kompetenzen oder Verantwortlichkeiten über die Durchführung von Aktionen an Agenten, also Organisationseinheiten oder Individuen, zuweisen. Sowohl Wagner (Wagner, 2002) als auch Rosenberg (Rosenberg und Dustdar, 2005) verzichten auf die Aufnahme dieser Kategorie in ihre Taxonomie von Regeltypen.

Fachliche Geschäftsprozessmodelle, besonders die, die auch als Arbeitsanweisungen in Unternehmen Anwendung finden, enthalten viele Informationen die auch als Business Rules aufgefasst werden können. Ein wichtiger Unterschied in der Ausdrucksweise zwischen Prozessen und Business Rules ist, dass der Geschäftsprozess beschreibt, *wie* etwas gemacht werden muss, während die Business Rules als deklaratives Statement beschreibt *was* gemacht werden muss (Rosenberg und Dustdar, 2005). Eine offensichtliche Abbildung von Regeln in

Prozessen findet sich unter anderem in Entscheidungsknoten, die entsprechend der Erfüllung einer Bedingung einen ausgehenden Pfad freischalten. Viele Autoren argumentieren, dass die Separation der Business Rules von den Geschäftsprozessen Vorteile hinsichtlich der Transparenz, Flexibilität und Effizienz bringt.

Transparenz, da die Anwendung von Business Rules auch die Definition und Einhaltung eines organisationsweiten, einheitlichen Begriffsglossars bedingt. Durch die Verwendung einheitlicher Begriffe, werden die Stellen, an denen sich die Veränderung einzelner Einflussgrößen auswirkt, leichter und eindeutig identifizierbar. Es bleibt anzumerken, dass auch eine idealtypische Prozessmodellierung auf einem einheitlichen Begriffsglossar basiert und somit die angestrebte Transparenz bieten kann.

Flexibilität, wird damit argumentiert, dass ein herausgelöstes Repository von Business Rules, einfacher gepflegt werden kann. Werden Regeln an mehreren Stellen wiederverwendet, so sinkt der Anpassungsaufwand weiter. Hierbei muss beachtet werden, dass sich die Regeln nur begrenzt inhaltlich, beispielsweise durch Wertanpassungen, ändern sollten. Bei strukturellen Änderungen in Regeln, wie die Hinzunahme weiterer Entscheidungsgrößen, müssen damit einhergehende Prozessänderungen beachtet werden. Generell bedeutet jede Separation von zusammengehörender Information, dass auch Mechanismen zur Re-Integration geschaffen werden müssen. Der dadurch notwendige Mehraufwand muss von den, durch die Separation entstehenden, Synergieeffekten gedeckt werden.

Effizienz, durch die Möglichkeit formal ausgedrückte Regeln von entsprechenden IT-Systemen automatisch auswerten zu lassen und damit die Transformation von modellierten Prozessen und zugehörige Regeln zu Ausführungssystemen wie Orchestrierungstools zu vereinfachen.

Abhängig vom Anwendungszweck und der Domäne kann der Einsatz von Business Rules in unterschiedlichen Abstraktionsebenen erfolgen. Wagner et al. (Wagner et al., 2004) unterscheiden drei Abstraktionsebenen gemäß der Model-Driven Architecture (MDA) der OMG:

- Auf Business Ebene, dem Computation Independent Model (CIM), werden Regeln und Geschäftsgrundsätze deklarativ, in natürlicher Sprache oder einer graphischen Modellierungssprache formuliert.
- Auf einer Plattform-unabhängigen Ebene, dem Platform Independent Model (PIM), werden Regeln in Formalismen oder computerverständlichen Mustern ausgedrückt, die in einem weiteren Schritt in einen ausführbaren Code transformiert werden können.

- Auf einer Plattform-spezifischen Ebene, den Platform Specific Models (PSM), werden die Regeln direkt in einer ausführbaren Sprache formuliert.

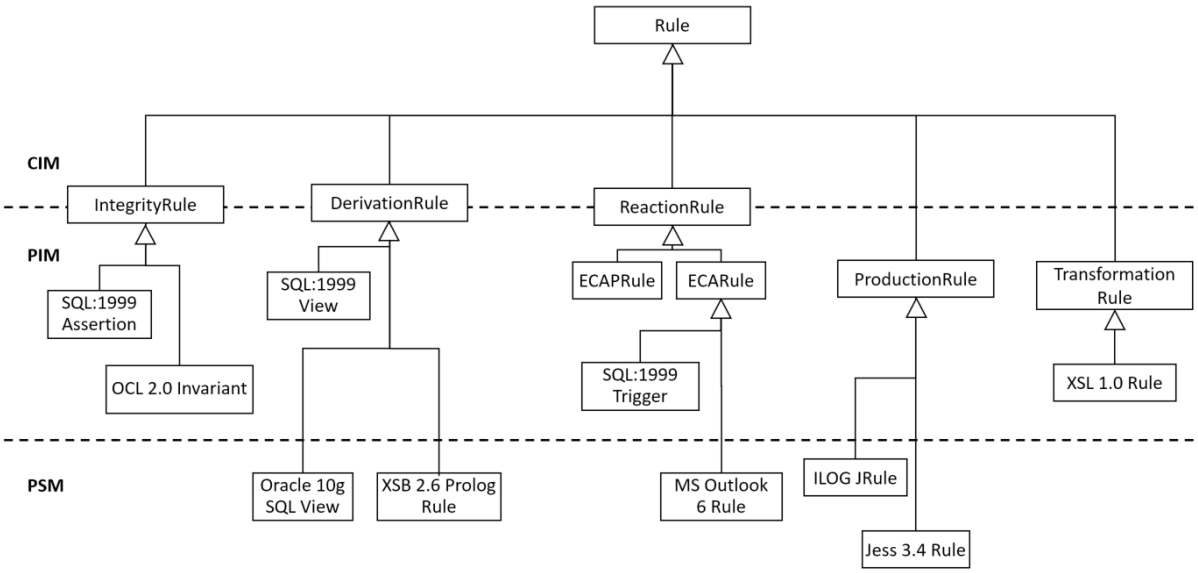


Abbildung 34: Abstraktionsebenen der Regelverwendung (Wagner et al., 2004)

Reaction Rule werden in (Taveter und Wagner, 2001) als der wichtigste Regeltyp bezeichnet. Die mit der Methode SIMchronization verbundenen Anforderungen bezüglich der Steuerung von Handlungen kann mit Reaction Rules umgesetzt werden. In früheren Aufsätzen von Wagner (Wagner, 2002) werden die Reaction Rules noch zu Production Rules spezialisiert. In Abbildung 34 gehören beide Regeltypen unterschiedlichen Klassifikationen an. Der maßgebliche Unterschied dabei ist, dass eine Reaction Rule aus einer Bedingung, die auf Events reagiert, einer optionalen zustandsabhängigen Bedingung (Vorbedingung), einem Action-Term und einem zustandsändernden Effekt oder Nachbedingung (Post-Condition) besteht. Sie wird deshalb auch Event-Condition-Action-Effect (ECAE) Rule oder Event-Condition-Action-Post-Condition (ECAP) Rule genannt. Die Production Rule hingegen besteht aus einer Untermenge von Termen, der zustandsabhängigen Bedingung (Vorbedingung) und dem Action-Term. Der Unterschied besteht also in der expliziten Berücksichtigung von Ereignissen auf der Bedingungs-Seite der Regel.

Auch wenn Production Rules nicht direkt von Ereignissen ausgelöst werden können, so kann man den Ereigniseintritt beispielsweise durch eine Veränderung eines Werts einer Variablen simulieren. Somit verhält sich eine Production Rule wie eine Reaction Rule. Auch eine Ableitungsregel kann mittels einer Production Rule nachgebaut werden indem die Aktion die Faktenbasis erweitert oder verändert (Wagner, 2005).

Durch die Flexibilität, die durch den Einsatz von Production Rules erreicht werden kann, werden Production Rules zur Abbildung des Verhaltens von Supply Chain-Akteuren und Prozessen in dieser Arbeit verwendet und im Folgenden näher erläutert.

2.8 Production Rules

Die Object Management Group, Inc. (OMG) definiert Production Rules folgendermaßen (OMG, 2009):

„… production rules provide an alternative, convenient representation for the many business rules that define the behavior (i.e., actions) in models and systems"

Eine Production Rule ist dabei ein Ausdruck, der bei der Erfüllung seiner Bedingung eine Aktion ausführt. Eine Bedingung kann mittels logischer Verknüpfung aus mehreren Teilbedingungen zusammengesetzt sein und die auszulösende Aktion kann aus einer Folge oder Liste von Einzelaktionen bestehen.

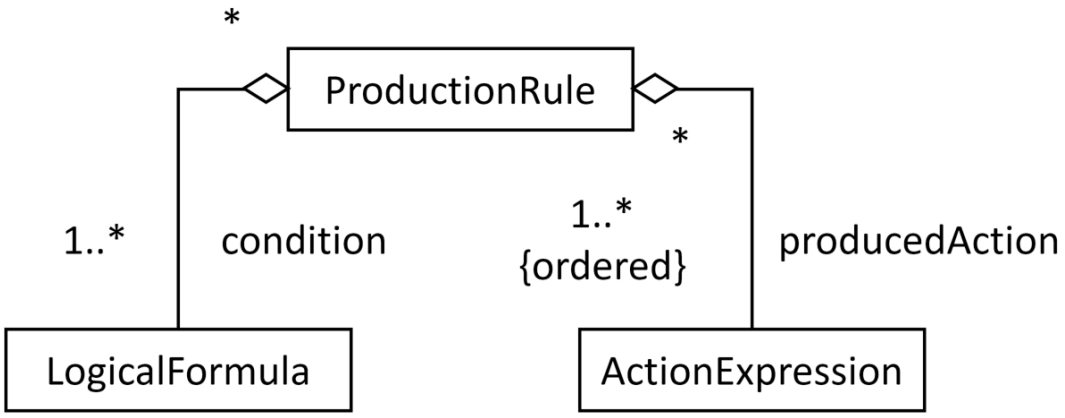


Abbildung 35: Das abstrakte Konzept der ProductionRules aus (Wagner, 2005)

Formulierung von Production Rules:

WENN [Bedingung] DANN [Aktion]

oder

WENN [Bedingung] DANN [Aktion] SONST [Alternative_Aktion]

Sollen mehrere Regeln zusammen verarbeitet werden oder lässt sich der Bezug zu einem fachlichen Prozess oder einer Aktivität nur unter Verwendung mehrerer Regeln herstellen, so können Regeln zu Regelsets gebündelt werden. Das Ergebnis eines Regelsets kann dabei von der Ausführungsreihenfolge der einzelnen Regeln abhängen.

Die Regeln eines Regelsets werden auf eine Datenbasis, Schacher et al. (Schacher und Grässle, 2006) sprechen von der Faktenbasis[50], angewendet. Diese Daten werden der Regel zum Zeitpunkt deren Ausführung als Parameter oder Kontext übergeben. Die Ausführung einer Production Rule resultiert in einem Ergebnis, das diese Datenbasis ändern kann.

Wagner (Wagner, 2005) hält fest, dass, obwohl Production Rules in den 80-er Jahren stark dazu verwendet wurden, um Expertensysteme zu implementieren, ihnen im Gegensatz zu Ableitungsregeln (wie in Prolog) eine fundierte und präzise theoretische Basis und formale Semantik fehlt.

Wichtig erscheint und die Unterscheidung zwischen Action und Aktivität (Taveter und Wagner, 2001). Während eine Action zu einem bestimmten Zeitpunkt erfolgt, wir eine Aktivität über eine Zeitdauer hinweg durchgeführt. Eine Action kann eine Aktivität auslösen, eine Aktivität kann aus mehreren Actions bestehen.

[50] Englisch „Fact base“

3. Anforderungen an eine Modellierungsmethode für Maintenance Supply Chains

Basierend auf den, in im vorherigen Kapitel beschriebenen Grundlagen und aufgeworfenen Problemstellungen, werden in diesem Kapitel Anforderungen an eine Modellierungsmethode für Maintenance Supply Chains gestellt. Wie in Kapitel 2.1.1 beschrieben, besteht eine Modellierungsmethode nach Karagiannis und Kühn (Karagiannis und Kühn, 2002) aus der Modellierungstechnik, den Auswertungsmechanismen inklusive Algorithmen und dem Vorgehensmodell. Bezüglich der Modellierungstechnik, also der Modellierungssprache oder Modellierungsnotation, muss die Frage beantwortet werden, ob eine bestehende Notation als Basis für die zu entwickelnde Methode herangezogen werden kann, eine bestehende Modellierungsnotation durch beispielsweise zusätzliche Objekttypen und -attribute erweitert werden kann oder ob eine gänzlich neue Modellierungsnotation entwickelt werden muss.

Zunächst werden grundsätzliche Anforderungen zu Modellierungsmethoden diskutiert.

3.1 Anforderungen zur Modellierungstechnik

Um eine Modellierungstechnik zu beschreiben, muss zum einen die Modellierungssprache definiert werden und zum anderen deren Handhabung festgelegt werden. Die Definition formaler Vorgaben, wie sie teilweise durch FDMM, im Kapitel 6 erfolgt, ermöglicht dem Benutzer die Erstellung semantisch ausdrucksvoller und syntaktisch korrekter Modelle. Korrekte Modelle sind schließlich die Grundvoraussetzung um Auswertungsalgorithmen, wie beispielsweise die Simulation, auf Modelle anwenden zu können und richtige und nachvollziehbare Ergebnisse zu erzielen. Neben den formalen Anforderungen an die Modellierungstechnik stellt Frank (Frank und Bodo, 2003) darüber hinaus noch anwenderbezogene Anforderungen an eine Modellierungssprache, wie intuitive Bedienbarkeit, und anwendungsbezogene Anforderungen, die sich aus der Domäne ergeben.

3.1.1 Formale Anforderungen

Auch für Frank (Frank und Bodo, 2003) sind die formalen Eigenschaften einer Modellierungssprache von herausragender Bedeutung, da sie die Voraussetzung für die erfolgreiche Nutzung von Modellen, wie zum Beispiel Anwendung von Analyseverfahren und Transformationsverfahren sind. Jedes automatische oder semi-automatische Verfahren, das auf die erstellten Modelle angewendet wird, setzt eine formale Korrektheit der Modelle voraus.

Eine grundlegende Anforderung an die Modellierungssprache ist, dass die von Becker, Rosemann, Schütte (Becker et al., 1995) aufgestellten Grundsätze ordnungsmäßiger Modellierung (GoM) von der Modellierungssprache unterstützt werden und sie den Benutzer in die Lage versetzt, Modelle zu erstellen, die den GoM genügen.

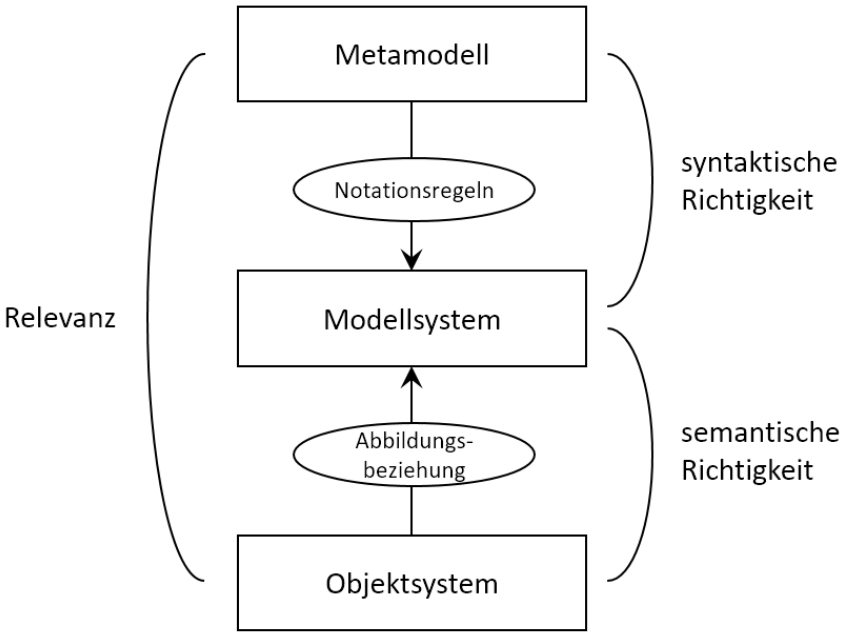


Abbildung 36: Abgrenzung der Grundsätze der Richtigkeit und Relevanz (Becker et al., 1995)

Sie unterscheiden dabei zwischen syntaktischer und semantischer Richtigkeit. Syntaktisch ist ein Modell korrekt, wenn es nur Informationsobjekte enthält, welche im Metamodell definiert sind und es die Notationsregeln einhält. Daraus ergibt sich für die, in dieser Arbeit anzuwendende Sprache die Anforderung, ein Metamodell bereitzustellen und Notationsregeln zu definieren, nach denen ein formal korrektes Modellsystem erstellt werden kann. Ein Modell ist semantisch richtig, sofern es die Struktur und, wenn erforderlich, das Verhalten des Objektsystem, also die Realität oder einen Zustand in der Zukunft, ausreichend genau abbildet. Die ausreichend genaue Modellierung wird aus dem Grundsatz der *Relevanz* abgeleitet, demgemäß der Inhalt des erstellten Modellsystems dem Modellierungsziel entsprechen muss. Für Becker, Rosemann, Schütte (Becker et al., 1995) ist die Relevanz einzelner Modellelemente dann gegeben, wenn durch Weglassen von Elementen der Nutzen der Modellverwendung verringert würde.

Brooks und Tobias (Brooks und Tobias, 1996) konstatieren, dass oft nur sehr vage Hinweise und Richtlinien über den richtigen Detailierungsgrad von Modellen gegeben werden, sie quotieren die Maxime des Philosophen William von Ockham der schrieb, „entities should not be multiplied without necessity“ und „it is vain to do by more what can be done by fewer“ und schlussfolgern, dass das einfachste Modell gewählt werden soll, das noch die Modellierungsziele erfüllt.

Den aufgestellten Grundsatz der Wirtschaftlichkeit des Modellierungsunterfangens kann nicht als eine Anforderung an die Modellierungstechnik oder Modellierungssprache gesehen werden, sondern muss durch geeignete Schritte im Modellierungsverfahren sichergestellt werden. Der Umfang eines Modellierungsprojekts und vor allem die Detaillierung der Modelle müssen zu Beginn des Projekts definiert werden und regelmäßig während der Durchführung überprüft und hinterfragt werden.

Weitere Grundsätze sind die Forderung nach Klarheit, welcher in Kapitel 3.1.1.2 behandelt wird, nach Vergleichbarkeit und nach einem systemischen Aufbau.

3.1.1.1 Vergleichbarkeit und Systemischer Aufbau

Der Grundsatz der Vergleichbarkeit ist in der zu entwickelnden Methode von besonderer Bedeutung, da eines der Einsatzszenarien der Vergleich von aktuell implementierten Supply Chains mit einem oder mehreren zukünftigen Alternativ-Szenarien ist. Da bei unternehmensübergreifenden Supply Chains die Teilmodelle der einzelnen Partner miteinander verglichen und im weiteren Verlauf deren Material- und Informationsflüsse angeglichen werden sollen, ist es wichtig, dass die Methode diesem Grundsatz unterstützt. Ähnlich gelagert ist die Forderung nach dem systemischen Aufbau. Das Metamodell der Methode soll unterschiedliche, für die Domäne und die typischen Fragestellungen wichtige Sichten unterstützen und integrieren und eine Komposition beziehungsweise Dekomposition von Objekten zulassen.

Laut Frank (Frank und Bodo, 2003) genügt die Spezifikation einer Sprache „dem Kriterium der Korrektheit, wenn sie die eindeutige Identifikation (syntaktisch und semantisch) unzulässiger Modelle gewährleistet und gleichzeitig erlaubt, die Menge aller zulässigen Modelle zu generieren“. Von einer vollständigen Sprache fordern sie, dass alle zur Verfügung gestellten Konzepte eindeutig bezüglich ihrer Verwendung definiert sind.

Für Junginger (Junginger, 2001) beschreiben die Knotentypen[51], Attribute und Strukturregeln die Syntax einer graphenbasierten Modellierungssprache, wie sie im hier dargestellten Ansatz verwendet wird. Besonders die Strukturregeln definieren syntaktisch korrekte Modelle und ihre Eigenschaften. Ein Beispiel für Strukturregeln sind Kardinalitäten von Relationen, wie sie im Kapitel 6 „Beschreibung des Metamodells in FDMM“ angegeben werden.

[51] Knotentypen werden in dieser Arbeit als Objekttypen oder auch Klassen bezeichnet

Aus den diskutierten Punkten ergibt sich die erste zentrale Anforderung:

Anforderung 1:
Es ist ein Metamodell bereitzustellen und Notationsregeln zu definieren, nach denen ein formal korrektes Modell erstellt werden kann.

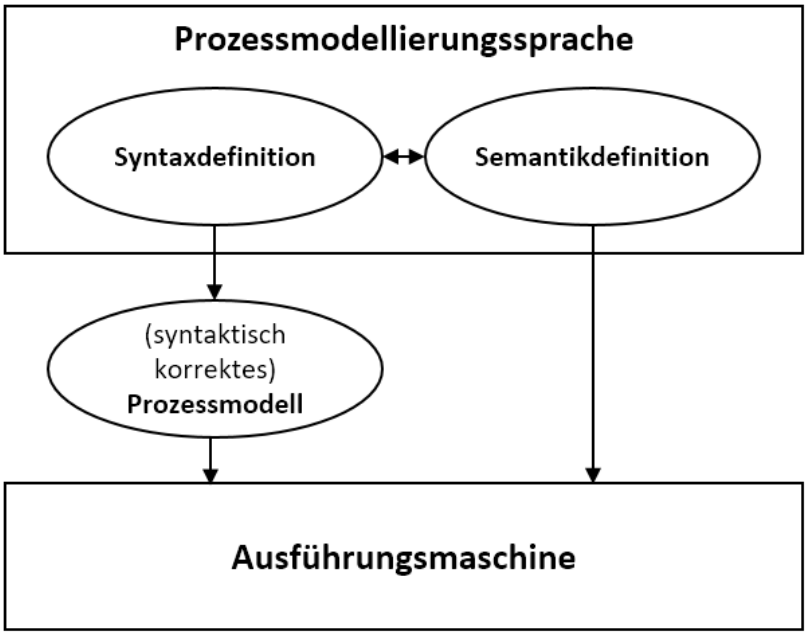


Abbildung 37: Grundkonzept einer Operationalen Semantikdefinition (Junginger, 2001)

3.1.1.2 Klarheit, Redundanzfreiheit und Wiederverwendung

Ähnlich wie der Grundsatz nach Klarheit in den GoM fordert Frank (Frank und Bodo, 2003), dass die von einer Modellierungssprache zur Verfügung gestellten Sprachkonstrukte einfach verständlich und intuitiv in ihrer Ausdruckskraft sein sollen. Die Sprache soll redundanzfrei sein, dies bedeutet, dass dem Benutzer genau ein Konzept angeboten wird, um einen semantischen Zusammenhang darzustellen. Diese Forderung, besonders im Zusammenspiel mit dem Grundsatz der Wirtschaftlichkeit ist ein starkes Argument für eine Modellierungssprache, welche die Konzepte, Begriffe und Zusammenhänge der Domäne, in der sie verwendet werden, aufgreift und genau die Objekte und Sichten anbietet, die der Modellierer benötigt, um effizient das Objektsystem in einem Modell abzubilden.

Die Forderung nach der Unterstützung von Wiederverwendung von Modellinhalten zielt auf eine Steigerung der Effizienz der Modellierung ab, da wiederkehrende Modellstrukturen nur einmal abgebildet werden müssen und über Referenzen in verschiedene andere Modelle eingebunden werden können. Außerdem können dadurch Modelle kleiner und besser

strukturiert werden, was wiederrum die Lesbarkeit und Verständlichkeit der Modelle erhöht. Aus dieser Forderung leitet sich die Notwendigkeit ab, dass die Sprache verschiedene Ebenen von Abstraktion unterstützt. Die Methode muss also unterschiedliche Detaillierungsgrade unterstützen, sich zum einen zur Abstimmung unternehmensübergreifender logistischer Prozesse eignen, aber zum anderen auch den Spezialisten die notwendige Transparenz und Instrumente zur Optimierung und Synchronisation der Material- und Informationsflüsse geben.

Angewendet auf die Domäne Supply Chain Management kann dies bedeuten, dass zum einen Modelle die unternehmensübergreifende Supply Chain End-to-End mit hoher Abstraktion abbilden und zum anderen deren Teilmodelle die unternehmensinternen Abläufe mit geringerer Abstraktion, aber dafür höherer fachlicher Aussagekraft, beschreiben.

Anforderung 2:

Die Modellierungssprache muss mehrere Ebenen von Abstraktion und die Wiederverwendung von Modellinhalten unterstützen.

3.1.2 Anwenderbezogene Anforderungen

Der in dieser Arbeit entwickelte Ansatz richtet sich in erster Linie an Praktiker in Unternehmen, welche zum einen die Zusammenhänge und Abhängigkeiten in der Maintenance Supply Chain besser verstehen wollen und zum anderen stetig mit der Verbesserung der Reaktionsgeschwindigkeit und generellen Effizienz der Abläufe befasst sind. Grundsätzlich lassen sich bei der Betrachtung der anwenderbezogenen Anforderungen zwei Nutzergruppen unterscheiden:

die Anwender der Methode: sie erstellen die Modelle, reichern sie mit Daten an, führen Analysen aus und präsentieren Ergebnisse. Diese Nutzer könnten beispielsweise interne Projektmitarbeiter, Berater oder IT-System-Integratoren sein und müssen über besondere Kenntnisse der Methode, ihre Anwendungsmöglichkeiten aber auch ihre Anwendungsgrenzen verfügen.

die Nutzer der Ergebnisse: sie übernehmen und interpretieren Ergebnisse oder Teilergebnisse der Analyse und wenden sie auf ihre fachliche Problemstellungen an. Diese Nutzer könnten beispielsweise Instandhalter, Logistiker oder generell Fachbereichsmitarbeiter sein und sollten schon bei der Erstellung der Modelle und ihrer Parametrisierung eingebunden sein, um die korrekte Abbildung des Objektsystems auf das Modell sicherzustellen.

Diese generellen Anwendergruppen können durch die Positionierung der Auswertungsmethode und die Fokussierung auf die Domäne Supply Chain Management für die Instandhaltung weiter detailliert werden und daraus Anforderungen pro Rolle abgeleitet werden.

Beispiele für Anwender der Methode sind:

Projektanalysten, zur Transparentmachung des Ist-Zustandes der Supply Chain und zur Unterstützung von Szenarioanalysen. Um den Istzustand mit einem zukünftigen Sollzustand zu vergleichen oder mehrere alternative Modelle gegeneinander abzuwägen. Zielkriterien können dabei wirtschaftliche Kenngrößen, wie die Erfüllung von Servicelevel-Vereinbarungen oder der Lagerbestand von Ersatzteilen sein. Des Weiteren können sie die Methode als Entscheidungshilfe in der Planungsphase einer Supply Chain und zur Bewertung von generellen Supply Chain Strategien (Kaczmarek, 2002) anwenden.

Qualitätsmanager, die Prozesse und Schnittstellen dokumentieren müssen, Zusammenhänge und Abhängigkeiten in der Supply Chain verdeutlichen wollen und die Modelle zur Kommunikation und Schulung von Mitarbeitern nutzen.

IT Architekten, um den Informationsfluss der Maintenance Supply Chain zu analysieren, und daraus die bestmögliche Lösungsarchitektur zu entwickeln.

Change Manager, um interne und externe Stakeholder auf die Veränderung vorzubereiten und die Vorteile der zukünftigen Struktur transparent und objektiv aufzuzeigen.

Sicherheitsbeauftragte, um im Falle einer Störung die betroffenen und zu informierenden Stellen anhand der Materialflüsse identifizieren zu können.

Beispiele für Nutzer der Methode sind:

Wartungstechniker, um eine einfache Beschreibung der Instandhaltungsschritte zu erhalten.

Instandhaltungs-Manager, um einen Überblick über die Leistungsfähigkeit und Elastizität (Störungsanfälligkeit) der Instandhaltung Supply Chain zu gewinnen.

Wartungsspezialisten, um eine Instandhaltungs- und Ersatzteillagerstrategie für verschiedene Anlageklassen definieren zu können.

Supply Chain Manager, um Informationsflüsse vom Kunden zu den Lieferanten über mehrere Stufen (Tiers) hinweg verstehen und optimieren zu können.

Produzenten von Anlagen, um über den Standort von Ersatzteillagern und die damit einhergehenden Reaktionszeiten entscheiden zu können.

Die initiale, im Rahmen dieser Arbeit entwickelten Modellierungsmethode wird nicht alle der oben aufgeführten Einsatzszenarien abdecken können. Aus dem breiten Feld von potentiellen Anwendern und damit einhergehend den sehr vielfältigen Einsatzgebieten für eine solche Methode ergibt sich jedoch die Anforderung, dass die entwickelte Methode erweiterbar sein muss, beispielsweise indem weitere Objekttypen und Attribute hinzugefügt werden. Schwermer (Schwermer, 1998) spricht in seinen Anforderungen zur Unternehmensmodellierung von der Anforderung der *Erweiterbarkeit von Modellen* in dem Sinne, dass das Modell für weitere, vielleicht dem ursprünglichen Modellierungszweck nicht entsprechenden, Anwendungen verwendbar sein sollte. Durch eine Erweiterungsmöglichkeit des zugrundeliegenden Metamodells kann diese Forderung zusätzlich unterstützt werden.

Anforderung 3:
Die Modellierungssprache muss erweiterbar sein und auf verschiedene Anwendungsszenarien hin anpassbar sein.

Eine weitere Anforderung, welche sich schon aus der formalen Anforderung an die Klarheit der Sprache ergibt, ist die Verwendung einer klar verständlichen, im Idealfall schon bekannten Notation und Begrifflichkeit (Prackwieser, 2013).

Anforderung 4:
Die Modellierungssprache muss über eine einfach verständliche und ausdrucksstarke Notation mit domänenspezifischer Begrifflichkeit verfügen.

3.1.3 Nicht-funktionale Anforderungen

Die Fülle der potentiellen Anwender und Nutzer der Methode legt nahe, dass die Umsetzung der Methode, ihrer Sprache und Algorithmen und Auswertungsmechanismen, möglichst benutzerfreundlich und intuitiv zu erfolgen hat. Ein wichtiges Erfolgskriterium ist dabei, dass die Methode effektiv anzuwenden ist und ihre Bedienungselemente in eine Anwendungsoberfläche integriert sind. Dies schließt die Verwendung von unterschiedlichen spezialisierten Tools nicht aus, um die benötigte Funktionalität der Methode bereitzustellen. Es muss aber sichergestellt werden, dass die Überführung, von Daten von einem Tool zum anderen, den Anwender nicht an der effizienten Nutzung der Methode hindert.

Anforderung 5:
Alle Bedienungselemente der Methode sollten in eine Anwendungsoberfläche integriert sein.

3.1.4 Anwendungsbezogene Anforderungen.

Im Kapitel 2.5.1 wurden die Vor- und Nachteile der graphischen Modellierung und der Simulation als Auswertungsverfahren gegenüber mathematischen, analytischen Verfahren diskutiert und die Verwendung einer graphischen Modellierungsmethode als Lösungsweg für die in dieser Arbeit behandelten Fragestellungen behandelt.

Aus der Zielsetzung dieser Methode, unternehmensübergreifende Supply Chains zu modellieren und dabei die Interaktion von Materialbewegungen und Informationsweitergaben zu untersuchen, ergeben sich prinzipielle Anforderungen an die Methode selbst und ihre Auswertungsmechanismen. Im Folgenden werden die Anforderungen an die Abbildungsfunktionalitäten von Material- und Informationsflüssen an die Methode gestellt. Darüber hinausgehende Flüsse, wie zum Beispiel finanzielle Transaktionen werden in dieser Arbeit nicht berücksichtigt[52].

3.1.4.1 Anforderung der Darstellungsmöglichkeit des Materialflusssystems

Offensichtlicher Zweck der Supply Chain ist es ein materielles Gut von einer Quelle, über beliebig viele Bearbeitungsstellen zu einem Empfänger zu bewegen. Die Möglichkeit der Darstellung dieser Kette von Tätigkeiten, unter Berücksichtigung der oben aufgeführten formalen Kriterien, ist die elementare Forderung an die Modellierungsmethode. Der Materialflusspfad besteht dabei aus einzelnen Stationen, welche Eigenschaften des Guts verändern, wie zum Beispiel es verpacken, bearbeiten oder seinen Standort verändern. Genau genommen handelt es sich nicht um einen durchgängigen und permanenten Fluss der zu einer stetigen Ortsveränderung führt, da Bearbeitungsstationen, wie Lager- oder Produktionsprozesse, zu Lager- und Wartezeiten führen können. Diese Stationen werden in Kapitel 4.2.1 als statische Objekte des Modells definiert.

[52] Finanzflüsse können zumindest für online Transaktionen als eine Spezialisierung des Informationsflusses angesehen werden. Wird eine online-transaktion durchgeführt, so entspricht dies einer Informationsweitergabe. Wird jedoch mit Geldscheinen bezahlt oder ein papierener Wechsel ausgestellt, könnte dies mittels eines Materialflusses abgebildet werden.

Die Methode muss Objekttypen zur Verfügung stellen, welche die Bearbeitung, die in diesen Stationen erfolgt, dem Grundsatz der Klarheit entsprechend, beschreiben können. Um die Sequenz des Materialflusses von Station zu Station zu definieren, werden gerichteten Konnektoren benötigt. In komplexeren Supply Chains kann der Materialfluss an bestimmten Stellen alternative Pfade durchlaufen, das heißt, dass sich der Fluss teilen und auch wieder zusammenführen lassen muss. Die Methode benötigt damit auch eine Möglichkeit die Bedingungen für die Wahl genau eines Pfades zu beschreiben.

In der Maintenance Supply Chain nimmt die instand zu haltende Anlage einen besonderen Stellenwert ein. Es wird dabei zwischen der Funktion die eine Anlage hat, also zum Beispiel „eine Flüssigkeit zu bewegen“ und der dafür installierten Einheit, die in diesem Fall eine Pumpe oder eine Archimedische Schraube sein kann, unterschieden. Die notwendige Funktion ändert sich üblicherweise nicht durch Instandhaltungstätigkeiten. Die dafür installierte Einheit kann aber sehr wohl temporär sein, da sie beispielsweise regelmäßig, im Rahmen der Wartung, ausgetauscht werden muss. Die Methode muss also die Funktion als statisches Objekt und die Einheit als dynamisches Objekt abbilden können.

Die in Kapitel 4.2.2 beschriebenen dynamischen Objekte stellen im Materialflusssystem die Güter, Ersatzteile oder wie eben beschrieben, instand zu haltende Einheiten dar. Sie sind die Objekte deren Aufenthaltszeit im Supply Chain System typischerweise minimiert werden soll oder deren Produktions- oder Logistikkosten zu minimieren sind. Sie traversieren entlang den angebotenen Materialfluss-Konnektoren durch das Modell, und werden von statischen Objekten einer Eigenschaftsveränderung unterzogen.

Die Methode muss diese Objekte als eigenständige, beobachtbare und untersuchbare Objekte behandeln und sollte deren Bewegung mittels einer Animation direkt im Modell anzeigen.

3.1.4.2 Anforderung der Darstellungsmöglichkeit des Informationsflusssystems

Zweck des Informationsflusssystems einer Supply Chain ist es, durch zielgerichtete und zeitlich abgestimmte Informationsweitergaben an betroffene Supply Chain Partner dafür zu sorgen, dass das Material schnellstmöglich, korrekt und kostengünstig durch die Supply Chain bewegt wird und beim Kunden in ausreichender Quantität und Qualität eintrifft. Sehr ähnlich zum Materialflusssystem werden statische Objekte, wie zum Beispiel Produktionsschritte, mit Auftragsinformationen versorgt oder Abrufe von Lagern getätigt. Neben den zuvor beschriebenen Objekten des Materialflusses muss es darüber hinaus noch ein Steuerungs- und Informationsverarbeitungsobjekt geben, das zentral oder dezentral den Materialfluss über den Informationsfluss steuern kann.

Wie im Materialflusssystem werden auch im Informationsflusssystem dynamische Objekte zwischen den statischen Objekten bewegt. Diese dynamischen Objekte sind hierbei

Informationsträger, wie zum Beispiel Nachrichten oder Abrufe, sie werden erzeugt, gesendet, empfangen und gelöscht. Sie müssen als eigenständige Objekte im Modell abbildbar und auswertbar sein. Dynamische Objekte existieren üblicherweise nur für eine relativ kurze Zeitspanne, da während der Betrachtungsdauer einer Analyse neue Nachrichten mit Botschaften entstehen, die für eine gewisse Zeit ihre Gültigkeit haben. Die Modellierungssprache muss Konzepte anbieten, mit denen das dynamische Verhalten dieser Objekte im Zeitverlauf oder als Reaktion auf Umwelteinflüsse definiert werden kann.

Die mit den statischen Objekten der Materialflusssystems als auch des Informationssystems verbundenen Tätigkeiten werden durch Ressourcen ausgeführt. Diese Ressourcen können beispielsweise Arbeitskräfte, Maschinen, Lagerstätten, Transportmittel oder Softwaresysteme sein. Es ist zwar nicht Ziel der in dieser Arbeit entwickelten Methode, eine Produktions- oder Ressourcenplanung durchzuführen, es sollte aber, um das Modell für die Zielgruppe verständlich zu machen, die Möglichkeit bestehen, die verantwortlichen Ressourcen zu annotieren.

Tabelle 3: Klassifizierung der Prozessdaten eines Intralogistikprozesses (Günthner und Schneider, 2011)

Hauptmerkmale eines intralogistischen Prozesses	**Ausprägungen der Hauptmerkmale eines intralogistischen Prozesses**
Ressourcen	Betriebsmittel und Hilfsstoffe
	Personal
	Flächen, Räumlichkeiten und Infrastruktur
Intralogistische Objekte	Material, Teile, Güter
	Information
Vorlagen und Aktivitäten	Beschreibung der Vorgänge
	Bewertung der Vorgänge
	Verantwortlichkeiten
	Restriktionen
	Sonstige

Günthner und Schneider (Günthner und Schneider, 2011, S. 24), siehe Tabelle 3, identifizieren als elementare Inhalte eines Supply Chain Modells, Ressourcen, Intralogistische Objekte[53] und Vorgänge und Aktivitäten sowie deren Ausprägungen. Sie bezeichnen diese Merkmale auch als notwendigen Prozessdaten. Dies entspricht der, in dieser Arbeit angestellten Forderung nach statischen Objekten zur Beschreibung von Vorgängen und Aktivitäten und Ressourcen.

[53] Der Begriff „Intralogistische Objekte“ ist bei der Betrachtung unternehmensübergreifender Supply Chain nicht adäquat

Die in diesem Ansatz beschriebenen Intralogistischen Objekte werden durch dynamische Objekte in SIMchronization abgebildet.

Anforderung 6:

In der Maintenance Supply Chain bearbeitete und transportierte materielle Teile und übertragene Informationen müssen im Modell individuell beschreibbar und auswertbar sein, um die dynamischen Zusammenhänge und Abhängigkeiten, zwischen Informations- und Materialfluss erkennen und deren Synchronisation verbessern zu können.

3.1.4.3 Verknüpfung von Informations- und Materialflusssystem

Die Verknüpfung von Informations- und Materialflusssystem ergibt das eigentliche Kontrollsystem der Supply Chain. Die Modellierungsmethode muss dabei sowohl die Modellierung von Supply Chain Netzen mit zentraler, dezentraler und autonomer Steuerung unterstützen. Unter einer autonomen Steuerung wird verstanden, dass materielle Güter Informationen tragen können, die sie beispielsweise zur nächsten Bearbeitungsstation führen können und die sie identifizierbar machen. Diese Auto-ID Verfahren gewinnen zunehmend an Bedeutung und müssen ebenfalls abbildbar sein. Als Anforderungen an die Modellierungssprache ergibt sich nun, dass die Möglichkeit bestehen muss, individuellen Teilen bestimmte Informationen zuweisen zu können, wie sie zum Beispiel in einem Barcode enthalten sind. Diese Information muss an definierten Stellen im Materialfluss auslesbar sein und die Information im Informationsflusssystem weiterverarbeitet werden können. Noch einen Schritt weiter gehen die Anforderungen zur Abbildung von RFID-Tags auf dynamischen Objekten des Materialflusses. Diese Tags können nur auslesbar sein oder aber auch beschreib- und lesbar. Dies bedeutet, dass das materielle Teil veränderliche Information tragen kann, also über einen Informationsspeicher verfügt und somit selbst zum Teil des Informationsflusses wird. Eine Aus- und Einlesestation durch die das Teil im Materialfluss geführt wird muss als Objekttyp vorhanden sein. Dieses Objekt ist eine Schnittstelle zwischen dem Material- und Informationsfluss. Für Günthner schließt RFID gar die Lücke zwischen Material- und Informationsfluss (Günthner und Schneider, 2011).

Anforderung 7:

Die Modellierungsmethode muss die Erstellung und Analyse von Modellen mit zentraler, dezentraler und autonomer Supply Chain Steuerung inklusive Auto-ID Verfahren unterstützen.

Ziel der Methode ist es, durch verbesserte Synchronisation, also Abstimmung von Informations- und Materialflüssen untereinander, die Supply Chain effizienter zu gestalten. Das zur Analyse durch den Nutzer verwendete Modell muss somit die im Materialfluss bewegten Objekte im Zusammenhang mit den auslösenden oder ausgelösten Informationsartefakten zeigen. Für den sehr wahrscheinlichen Fall dass eine Information oder ein Ereignis eine Kette von Aktionen und Reaktionen auslöst, muss die Sequenz der Zustandsänderungen durch Nummerierung der dynamischen Objekte, also beispielsweise Teile und Nachrichten unterstützen.

3.1.4.4 Anforderungen an die Simulation

Um die dynamischen Abläufe und Zustandsänderung innerhalb des Supply Chain Netzwerks nachvollziehen und analysieren zu können, muss das Modell simulierbar sein. Das bedeutet zum einen, dass das Metamodell alle Klassen und Attribute zur Verfügung stellen muss, die für eine Simulation erforderlich sind und zum anderen, dass ein Simulationsalgorithmus existieren muss, der auf diese Modelle angewendet werden kann. In der Domäne Instandhaltung und im Speziellen in der, die Instandhaltungsmaßnahmen versorgenden Supply Chain, erfolgen die meisten Veränderungen oder Transaktionen zu diskreten Zeitpunkten, stetige Zustandsänderungen spielen hierbei eine untergeordnete Rolle[54]. Demgemäß ist ein diskreter Simulationsalgorithmus notwendig, um die Dynamik und Konsequenzen der Zustandsänderungen analysieren zu können.

Hauptanforderung an die Simulation in diesem Ansatz ist, dass das an sich statische Modell im Zeitverlauf dynamisiert wird, um dabei das Verhalten der Systemkomponenten studieren und auswerten zu können. Die Simulation muss die im Metamodell definierten material- und informationsflussorientierten Objekttypen unterstützen, deren Semantik im jeweiligen Kontext richtig interpretieren und zur Ausführung bringen.

Anforderung 8:
Die Methode muss eine diskrete ereignisorientierte Simulationskomponente beinhalten, die in der Lage ist, sowohl Material- als auch Informationsflüsse der Supply Chain im Zeitverlauf zu simulieren.

Jedes individuelle, im Materialfluss bewegte, Teil und jede im Informationsfluss versendete Nachricht oder jeder Abruf muss von der Simulation als ein einzelnes beobachtbares und

[54] Ein stetiger Prozess ist der andauernde Abbau des Abnutzungsvorrats von operativen Teilen. Dieser wird im Kapitel 2.4.1.1 behandelt.

auswertbares Element betrachtet werden. Das ermöglicht die stückgenaue Analyse der Lagermengenentwicklung und die Identifikation von Engpässen für spezielle Ersatzteile.

Die Modellierung muss einen, für die Simulation verständlichen, Mechanismus bereitstellen, um folgende Parameter oder deren Veränderung zu beschreiben:

- Ereignisse, wie Maschinenausfälle und Störungen der Lieferkette, sind nur schwer prognostizierbar und ihre Eintrittsmenge und Eintrittszeitpunkte unterliegen einer Zufallsverteilung. Um diese Einflüsse in einer Simulation berücksichtigen zu können, müssen diese Ereignisse, im speziellen deren Eintrittsmenge, Eintrittszeitpunkt, Ausprägung und Konsequenz, im Modell definierbar sein. Eine Nebenanforderung ist, dass ein Zufallszahlengenerator zur Verfügung stehen muss, um eine stochastische Simulation zu ermöglichen.
- Die Ausführungszeit derselben Produktionsschritte, Instandhaltungsmaßnahmen oder Transporte kann je nach Ausprägung der Prozessinstanz oder in Abhängigkeit von äußeren Einflüssen für jedes Teil unterschiedlich sein. Beispielsweise kann ein Lerneffekt in der Durchführung eines Produktionsvorgangs eintreten der dazu führt, dass mit fortschreitender Simulationszeit die Bearbeitung schneller erfolgt. Die sich daraus ergebende Anforderung ist, dass die Bearbeitungszeiten für Produktions- und Transportprozesse während der Simulation veränderbar sein müssen und diese Veränderung mittels einer Regel definiert werden kann.
- Der Weg, den ein Ersatzteil durch die Maintenance Supply Chain nimmt, kann sich abhängig von bestimmten Einflussfaktoren ändern. Das heißt, dass zwei Teile des gleichen Typs unterschiedliche Materialflusskanäle benutzen können oder müssen. Beispielsweise können dringend benötigte Ersatzteile über einen schnelleren Transportweg bewegt werden als Teile desselben Typs, die in geplanter Weise ein Zwischenlager auffüllen. Je nachdem woher die Information zur Steuerung des Teils durch die Supply Chain kommt, ergeben sich daraus zwei Anforderungen.
    - Eine zentrale Steuerung von Teilen an Materialfluss-Verzweigungspunkten muss simulierbar sein und die dafür gültigen Bedingungen definierbar sein.
    - Sollte das Teil selbst seinen Weg, beziehungsweise die notwendigen Bearbeitungsschritte, durch die Supply Chain kennen, beispielsweise gespeichert auf einem Smarttag, so muss diese Information von der Simulation auslesbar, auswertbar und in einer korrekten Steuerung des Teils umsetzbar sein.
- Um den stetigen Abbau des Abnutzungsvorrats einer Einheit definieren zu können, muss eine Funktion, welche diese Abbaukurve abbildet, formelhaft einer Lokation, in der die Zustandsänderung des Teils eintritt, zugeordnet werden können.

Anforderung 9:
Dynamische Veränderungen des Modells können durch ein mächtiges Regelwerk definiert werden, das während der Simulation ausgewertet wird und den Verlauf der Simulation beeinflussen kann.

Die Simulation muss sein während eines Durchlaufs Daten sammeln, welche den Benutzer in die Lage versetzen, mittels der Auswertungskomponente folgende Kennzahlen zu ermitteln (Kaczmarek, 2002):

Zeiten:

- Durchlaufzeiten, wie zum Beispiel der Zeit vom Eintreffen eines Kundenauftrags bis zur Anlieferung des Produkts beim Kunden
- Bearbeitungszeiten der verschiedenen Produktionsschritte pro Teil
- Wartezeiten pro Teil und Bearbeitungsstation. Wartezeiten sind die Zeiten, die ein Teil in einem Lager verbringt und auf den Start in der nächsten Bearbeitungsstation wartet
- Reaktionszeiten der Supply Chain auf Nachfrageänderungen oder Störungen
- Lagerumschlagsdauer, wie lange im Durchschnitt ein Teil in einem bestimmten Lager liegt

Mengen:

- Produktions-/ Liefermengen in einer definierten Zeitdauer
- Lagerbestände pro Zeiteinheit
- Nachfragemenge pro Produkt/Kanal
- Liefermenge pro Produkt/Kanal
- Anzahl der erteilten Produktionsorder / Arbeitsaufträge
- Anzahl der nicht ausgeführten Produktionsorder / Arbeitsaufträge, da zum Beispiel ein Ersatzteil fehlte

Qualität:

- der Verlauf der Zustandsabbaukurve einer Einheit in einer Lokation
- Auswertung über die Lebensdauer eines Teils und seiner Bestandteile

3.1.4.5 Anforderung an eine Animationskomponente

Das vom Modellierer erstellte graphische Supply Chain Modell bildet die statischen Komponenten der Supply Chain ab. Es enthält unter anderem die Planungs-, Liefer-, Lager- und Bearbeitungsprozesse und deren Verknüpfungen im Material- und Informationsfluss. Der im vorherigen Kapitel geforderte Mechanismus zur Definition des dynamischen

Verhaltens des Modells im Zeitverlauf, muss als additive Komponente zum statischen Modell gesehen werden. Dieser Mechanismus gibt vor, wann und wie die einzelnen dynamischen Objekte, also Teile oder Informationen, im Modell erzeugt, bewegt, verändert und verbraucht werden und welche Aktionen und Reaktionen von statischen Objekten durch sie ausgelöst werden. Das Verständnis des Verhaltens dieser dynamischen Objekte ist elementar für das Erkennen der zeitlichen Zusammenhänge und Abhängigkeiten in der Supply Chain und schlussendlich zur Verbesserung der Synchronisation von Material- und Informationsfluss.

Der Animation fällt dabei die Aufgabe zu, die dynamischen Objekte sichtbar zu machen und ihre Bewegung durch das Modell dem Modellnutzer zu verdeutlichen. Dabei sollten die Objekte direkt in dem, vom Modellierer erstelltem und vom Fachbereich verstandenem, Modell visualisiert werden. Eine Visualisierung in einem aus dem ursprünglichen Modell abgeleiteten Modell, das eigens für die Simulation generiert wurde, ist sub-optimal.

Anforderung 10:
Die Animationskomponente muss dynamische Objekte wie Teile, Nachrichten und Abrufe direkt im Supply Chain Modell darstellen und deren Bewegung durch das Modell visualisieren können.

Die Animation kann live, also direkt während der Simulation durchgeführt werden oder nach Beendigung der Simulation als Playback abgespielt werden (siehe Kapitel 0 „

Animation“). Beide Varianten haben Vor- und Nachteile die sich vor allem bei längeren Simulationsläufen und stochastischen Eingangsparametern auswirken. Eine Live-Animation zeigt Probleme, wie unbegrenzt wachsende Lagerstände, während der Simulation sofort auf und ermöglicht es so dem Nutzer die Simulation abzubrechen, das Modell zu korrigieren und neu zu starten.

In einer Animation, die nach der Simulation durchgeführt wird, können Phasen geringer Zustandsänderungen im Modell übersprungen werden und kritische Momente beliebig oft wiederholt beobachtet werden.

Anforderung 11:
Die Animationskomponente sollte während der Simulation dem Nutzer Feedback über den Verlauf der Simulation geben und eine Playback-Animation nach der Simulation erlauben.

3.1.4.6 Anforderungen an das Reporting

Um die Ergebnisse der Simulation und die Beobachtungen aus der Animation weiter analysieren und dokumentieren zu können, muss eine Funktionalität zur Verfügung stehen, die einen Export der während der Simulation gesammelten Daten aus dem System ermöglicht.

Offensichtlich wäre eine Vielzahl verschiedener Berichte, die alle Anforderungen der in Kapitel 3.1.2 (Seite 87) aufgelisteten Nutzergruppen und deren Einsatzszenarien unterstützen, wünschenswert. Da sich aber je nach individuellen Nutzer und Einsatzzweck die notwendigen Berichte unterscheiden können und sich mit fortschreitender Einsatzbreite der Methode auch die Reports weiterentwickeln werden, ist die zentrale Anforderung an eine solche Funktionalität, dass alle relevanten Daten in ein für den betrieblichen Nutzer weiterverwendbaren Format exportiert werden können. Dies ist typischerweise ein Tabellenkalkulationsprogramm, wie Microsoft Excel oder ein Datenbanksystem wie Microsoft Access.

Neben der Ausgabe eines solchen Protokoll- oder Log-Files sollten zentrale Ausgabewerte auch direkt im Modell als Graphen ausgegeben werden können. Diese Darstellung hat den Vorteil, dass die Entwicklung von Veränderungen direkt am Ort ihres Entstehens oder ihrer Auswirkung im Modell visualisiert werden können. Für den Nutzer der Ergebnisse kann damit sehr transparent das Verhalten des Systems und die Entwicklung wichtiger Kenngrößen dargestellt werden.

Neben statischen Reports, ist eine Darstellung der Sequenz von Aktionen und den dadurch ausgelösten Reaktionen wichtig, um die Abhängigkeiten in der Supply Chain verstehen und dokumentieren zu können. Offensichtlich ist die Animation ein geeignetes Werkzeug, um dies darzustellen, muss aber ein Dokument, wie zum Beispiel eine Trainingsanleitung, oder ein Business Case erstellt werden, so ist eine statische Abbildung dieser dynamischen Zusammenhänge gefordert.

Beispiele für Modellierungssprachen mit denen solche Sequenzen abgebildet werden können sind UML Sequenzdiagramme, siehe Abbildung 38, und Kooperationsbilder, siehe Abbildung 39. Sequenzdiagramme zeigen dabei eine Interaktion an, die durch eine Sequenz von Meldungen, also Informationsweitergaben, zwischen Objekten, Komponenten, Sub-Systemen oder Akteuren gebildet wird. Vertikale, gestrichelte Linie sind sogenannte Lebenslinien auf denen Vorkommnisausführungen, im Sprachgebrauch dieser Arbeit, die Prozesse oder Tätigkeiten, platziert werden. Horizontale Pfeile bilden Meldungen zwischen diesen Vorkommnissen ab. Die Zeit verläuft in Sequenzdiagrammen von oben nach unten, somit ergibt sich die Reihenfolge der Meldungen aus der Positionierung der Pfeile auf der y-Achse.

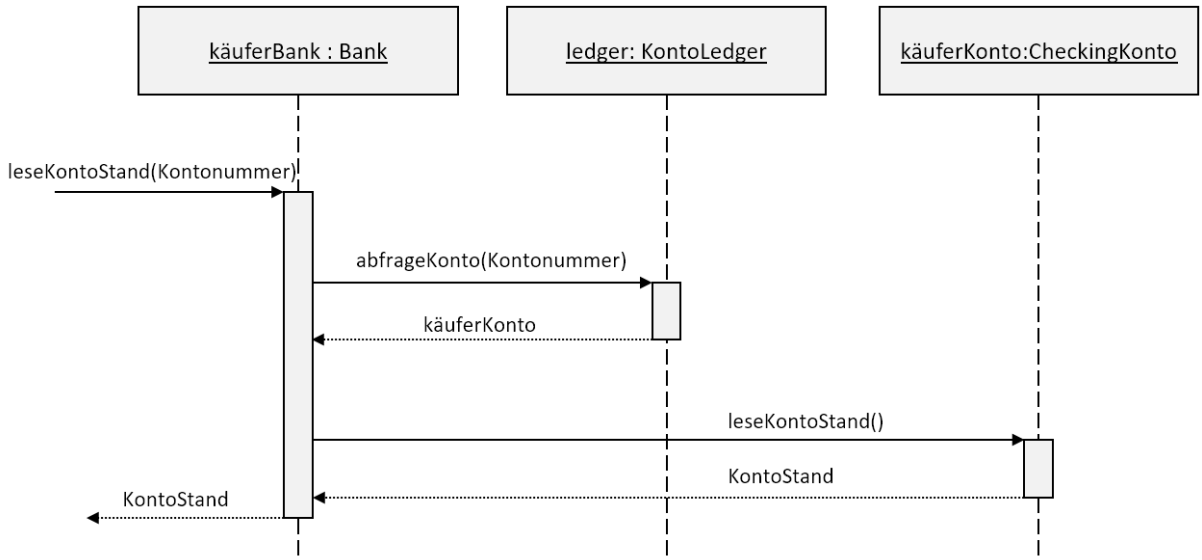


Abbildung 38: Beispiel eines Sequenzdiagramms

Während Sequenzdiagramme[55] den Informationsfluss abbilden und hauptsächlich zur Spezifikation von Softwaresystemen verwendet werden, erlauben die Kooperationsbilder neben der Abbildung der Interaktion auch die Darstellung weitergegebener Gegenstände. Kooperationsbilder stellen alle an einem Prozess beteiligten Akteure dar. Die Pfeile zwischen den Akteuren sind Kooperationen, die entweder eine Informationsweitergabe oder den

[55] Und die ähnlichen UML Kommunikationsdiagramme

Transfer von realen Objekten darstellen. Die Reihenfolge dieser Kooperationen ergibt sich aus der Nummerierung der Pfeile.

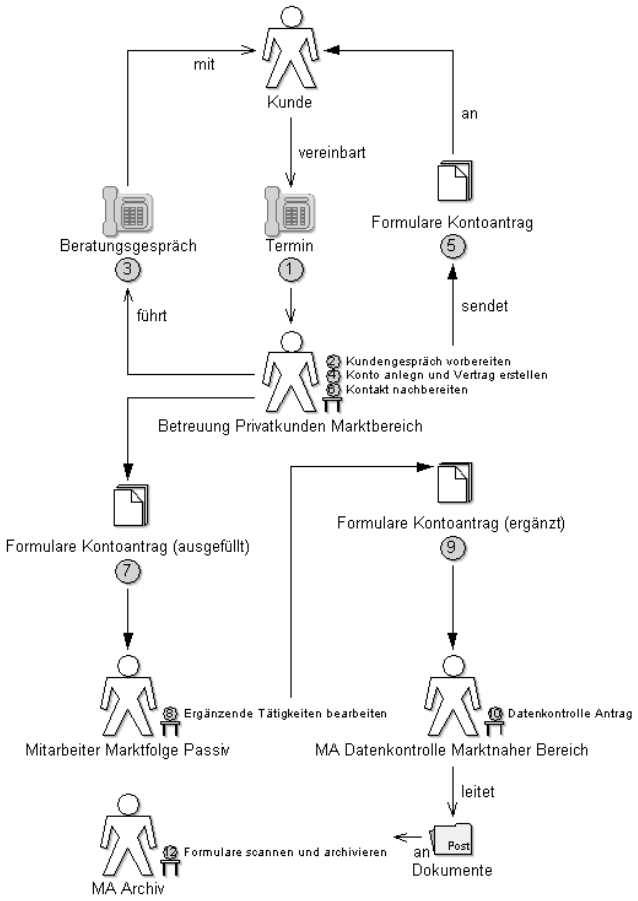


Abbildung 39: Beispiel eines Kooperationsbilds

Beide Modellierungsmethoden bilden nur ein bestimmtes Szenario ab, sie verfügen also über keine Notationselemente zur Beschreibung alternativer Pfade. Ihr Fokus ist die Modellierung exemplarischer, konkreter Szenarien ohne auf Fallunterscheidungen in einem Modell einzugehen (Breitling et al., 2006). Da die Abbildung verschiedener Pfadalternativen in einem Modell die Graphik überladen würde und der Vergleich von Modellalternativen auch durch Nebeneinanderlegen der Modelle möglich ist, kann die aus der Simulation resultierende Beschreibung der Zustandsfolge exemplarisch sein, also nur einen Fall abbilden.

Anforderung 12:
Eine Beschreibung der Zustandsfolge eines Modells von einem Startzeitpunkt bis zu einem Endzeitpunkt sollte als Bericht generierbar sein.

3.1.4.7 Zusammenfassung des Kapitels

In diesem Kapitel wurden die Anforderungen an die Methode SIMchronization im Detail diskutiert und diese in 12 zentrale Anforderungen zusammengefasst.

Tabelle 4: Anforderungen an die Methode SIMchronization

Nummer	Anforderungsbeschreibung
1	Es ist ein Metamodell bereitzustellen und Notationsregeln zu definieren, nach denen ein formal korrektes Modell erstellt werden kann.
2	Die Modellierungssprache muss mehrere Ebenen von Abstraktion und die Wiederverwendung von Modellinhalten unterstützen.
3	Die Modellierungssprache muss erweiterbar sein und auf verschiedene Anwendungsszenarien hin anpassbar sein.
4	Die Modellierungssprache muss über eine einfach verständliche und ausdrucksstarke Notation mit domänenspezifischer Begrifflichkeit verfügen.
5	Alle Bedienungselemente der Methode sollten in eine Anwendungsoberfläche integriert sein.
6	In der Maintenance Supply Chain bearbeitete und transportierte materielle Teile und übertragene Informationen müssen im Modell individuell beschreibbar und auswertbar sein, um die dynamischen Zusammenhänge und Abhängigkeiten, zwischen Informations- und Materialfluss erkennen und deren Synchronisation verbessern zu können.
7	Die Modellierungsmethode muss die Erstellung und Analyse von Modellen mit zentraler, dezentraler und autonomer Supply Chain Steuerung inklusive Auto-ID Verfahren unterstützen.
8	Die Methode muss eine diskrete ereignisorientierte Simulationskomponente beinhalten, die in der Lage ist, sowohl Material- als auch Informationsflüsse der Supply Chain im Zeitverlauf zu simulieren.
9	Dynamische Veränderungen des Modells können durch ein mächtiges Regelwerk definiert werden, das während der Simulation ausgewertet wird und den Verlauf der Simulation beeinflussen kann.
10	Die Animationskomponente muss dynamische Objekte wie Teile, Nachrichten und Abrufe direkt im Supply Chain Modell darstellen und deren Bewegung durch das Modell visualisieren können.
11	Die Animationskomponente sollte während der Simulation dem Nutzer Feedback über den Verlauf der Simulation geben und eine Playback-Animation nach der Simulation erlauben.
12	Eine Beschreibung der Zustandsfolge eines Modells von einem Startzeitpunkt bis zu einem Endzeitpunkt sollte als Bericht generierbar sein.

Das in den nachfolgenden Kapiteln beschriebene Konzeptionelle Modell, das Metamodell und die prototypische Implementierung basieren auf diesen Anforderungen. Im Rahmen der Evaluation wird bewertet, inwieweit die Methode und die prototypische Implementierung diese Anforderungen erfüllen.

4. Konzeptionelles Modell

4.1 Einleitung

Gemäß dem OMiLAB Life Cycle (siehe Kapitel 2.2.3) wird, nachdem die Domäne und ihre speziellen Anforderungen verstanden sind, in der Create Phase ein konzeptionelles Modell der Methode entwickelt. Das in diesem Kapitel entwickelte Modell, strukturiert die Lösung in ihre Komponenten, beschreibt das Zusammenspiel dieser Komponenten und die resultierenden Outputs, siehe Abbildung 40. Die Modellierungssprache selbst wird im konzeptionellen Modell noch nicht näher betrachtet, da sich aus den erforderlichen Funktionalitäten der Methode noch detailliertere Anforderungen an Modellierungsklassen und im speziellen deren Attributierung ergeben werden.

Eine der zentralen Designentscheidungen der Modellierungsmethode für SIMchronization ist die Abbildung der Dynamik der Supply Chain auf eine Art und Weise, die es dem Modellnutzer ermöglicht, die Zusammenhänge und Abhängigkeiten zwischen und innerhalb der Sub-Systeme Informationsfluss und Materialfluss zum einen zu erkennen und zum anderen zu analysieren, um eine Verbesserung der Synchronisation zu erzielen. Dies setzt voraus, dass neben den, die Supply Chain bildenden Objekte, auch die Objekte, die als Information oder Material durch die Supply Chain fließen, im Modell sichtbar sind. Offensichtlich ändern diese Objekte im Zeitverlauf ihre Lokation oder andere Eigenschaften. Werden nun diese, im Folgenden als dynamisch, bezeichneten Objekte als Teil des Modells verstanden, so müsste für jeden Zeitpunkt einer Zustandsänderung ein neues Modell erstellt werden. Dies ist zum einen vom Modellierungsaufwand gesehen nicht praktikabel und zum anderen, speziell bei größeren stark vernetzten Supply Chains sehr schwierig, da womöglich nicht alle Zustandsänderungen erkannt und im Modell abbildet werden können.

Um dennoch dem Modellnutzer eine sequentielle Abfolge der Modellzustände präsentieren zu können, werden anstatt die Auswirkungen der Zustandsänderungen, wie beispielsweise bewegte Materialobjekte, die Ursachen und Reaktionen, die eine Zustandsänderung bewirken, im Modell abgebildet. Eine Funktionalität des verwendeten Modellierungstools wertet diese Ursachen aus und verändert entsprechend des Resultats die dynamischen Objekte des Modells. Die dafür benötigte Funktionalität ist ein Simulationsalgorithmus und zur Visualisierung der Ergebnisse eine Animationsfunktion.

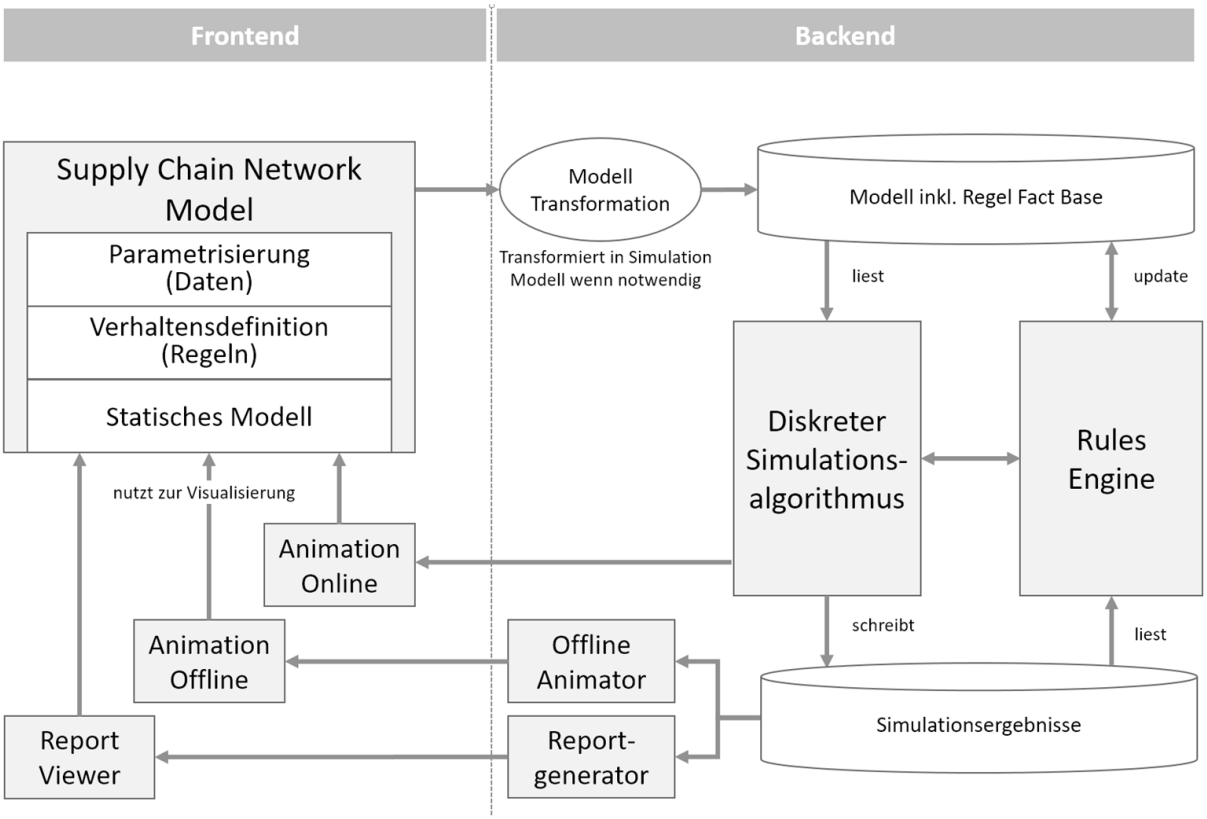


Abbildung 40: Konzeptionelles Modell der Methode SIMchronization

4.2 Modellierungskomponente

Wie Objekte des Material- und Informationsflusses durch die Supply Chain bewegt werden und wie das System auf Ereignisse reagiert, kann als Systemverhalten bezeichnet werden. Bricht man das System in seine Einzelteile auf, so kann für jede Komponente ein individuelles Verhalten definiert werden. Es werden zwei Arten von Objekttypen unterschieden.

4.2.1 Statische Objekttypen

Von ihnen abgeleitete Objekte repräsentieren Komponenten der Supply Chain, wie Prozesse, die über den gesamten Betrachtungszeitraum des Modells vorhanden sind und deren Lokation sich im Material- und Informationsfluss nicht ändert. Statische Objekte werden nicht durch Tätigkeiten im Modell erzeugt oder konsumiert, sie führen Bearbeitungen im Sinne ihrer Bestimmung durch. Dazu gehören beispielsweise Produktions-, Planungs-, Beschaffungs- und Lagerprozesse. Diese Prozessschritte werden von Ressourcen, wie Maschinen, Transportbänder, Mitarbeiter in speziellen Rollen und IT-Systemen durchgeführt. Auch wenn diese Objekte permanent im Modell existieren, bedeutet das nicht, dass diese Objekte ständig und unbegrenzt für die Bearbeitung neuer Aufträge zur Verfügung stehen,

denn sie können beispielsweise durch vorhergehende Aufträge belegt sein oder erst ab einem bestimmten Zeitpunkt aktiv werden. „Statisch“ heißt auch nicht, dass sich ihr Verhalten während der Betrachtungsdauer des Modells nicht ändern kann. Je nach Zustand des Systems kann ein statisches Objekt unterschiedlich reagieren.

Ein Modell, das ausschließlich aus statischen Objekten besteht wird als Strukturmodell der Supply Chain bezeichnet.

4.2.2 Dynamische Objekttypen

Dynamische Objekte werden zu einem Zeitpunkt im Betrachtungszeitraum generiert oder treten in das beobachtete System ein und ändern typischerweise ihre Lokation im Material- oder Informationsfluss im Zeitverlauf. Werden sie durch ein statisches Objekt bearbeitet, so kann sich ihr Zustand ändern. Sie werden im Material- oder Informationsfluss durch statische Objekte im Modell bewegt und schlussendlich verbraucht. Zu den dynamischen Objekten zählen die beweglichen logistischen Objekte, also Material oder Teile und Informationsobjekte wie Nachrichten und Abrufe.

Liebl (Liebl, 1992, S. 88) unterscheidet in seinem theoretischen Bezugsrahmen zu Bestandteilen diskreter Systeme ebenfalls zwei Klassen, die der als „permanent entities“ und „temporary entities“[56] bezeichnet.

4.2.3 Beeinflussung Dynamischer Objekte durch Statische Objekte

Die zu entwickelnde Modellierungssprache zur Abbildung der Maintenance Supply Chain wird, um Anforderung 6 zu erfüllen, verschiede Objekttypen mit jeweils unterschiedlicher Semantik bereitstellen. In vielen Modellierungssprachen ist die Semantik einzelner Objekttypen unveränderbar festgelegt, manchmal kann die Bedeutung eines Objekts durch Veränderung eines Objektattributs zum Zeitpunkt der Modellerstellung verfeinert oder genauer spezifiziert werden. Ein Beispiel für die genauere Spezifizierung ist das Attribut „Bearbeitungszeit“ des Objekttyps „Aktivität“ in ADONIS BPMS. Dieses verhaltensbeeinflussende Attribut des Objekts kann nur vor Beginn der Simulation als Konstante festgelegt werden und während der Ausführung nicht beeinflusst werden.

Die Methode SIMchronization benötigt eine Möglichkeit der Spezifizierung des Verhaltens von statischen Objekten, die es erlaubt, auch während der Simulation je nach Modellzustand unterschiedlich auf Aktionen zu reagieren und somit dynamische Objekte zu beeinflussen.

[56] Man findet für „temporary entities“ auch die Begriffe „transitory entities“ und „transient entities“

Um eine weitgehende Flexibilität bei der Definition des Verhaltens von Modellierungsobjekten zu erzielen, werden die auslösenden Aktionen und darauf folgenden Reaktionsmuster für jedes Objekt des Modells individuell mittels einer Regelsprache definiert. Vasilecas et al. (Vasilecas et al., 2016) folgern aus ihrer Literaturstudie, dass Geschäftsregeln verwendet werden können, um die Dynamik von Prozessen zu beschreiben. Leutgeb et al. (Leutgeb et al., 2007) beschreiben die semantische Anreicherung von Geschäftsprozessen um eine höhere Transparenz, Flexibilität und Effizienz der Prozesse zu erreichen.

Die Angabe dieses Regelsets erfolgt in speziellen Attributen der statischen Objekte. Auslösende Ereignisse können unter anderem, das Eintreffen eines dynamischen Informationsobjekts, ein Produktionsauftrag gemäß Produktionsplan oder ein externes Ereignisses sein.

Das Modell, das vom Modellersteller definiert werden muss, besteht somit aus einem statischen Modell der Supply Chain inklusive verhaltensbeeinflussender Regeln. Die Ausführung der Regeln führt zu Zustandsänderungen, die eine Bearbeitung und einen Transport von dynamischen Objekten zur Folge haben kann. In dem von Petsch et al. (Petsch et al., 2007) verfolgtem Ansatz werden Regeln dazu verwendet, um in agentenbasierten Systemen, die Lenkung des individuellen Verhaltens und die Zielsetzung einzelner Akteuren zu beschreiben.

4.3 Simulationsalgorithmus

Hauptaufgabe des Simulationsalgorithmus ist die Dynamisierung des statischen Modells inklusive dessen verhaltensbeschreibenden Regeln. Der Simulationsalgorithmus bewegt die Simulationsuhr durch die vom User definierte Betrachtungsperiode vorwärts und arbeitet dabei sequentiell alle auftretenden Ereignisse ab. Betrifft ein Ereignis ein statisches Objekt, so liest der Algorithmus die verhaltensbeschreibenden Regeln des Objekts aus den Objektattributen aus und leitet sie zur Auswertung an die Rules Engine weiter. Entsprechend den von der Rules Engine zurückgelieferten Ergebnissen werden Aktionen gesetzt und gegeben falls neue Ereignisse in der Zukunft generiert.

Die während eines Simulationslaufs generierten Ergebnisse werden unmittelbar in einer Datenbank gespeichert.

Die diskrete Simulation benötigt folgende Inputs.

- das statische Modell der Supply Chain
- das Ergebnis der Auswertung verhaltensbeschreibender und ereignisdefinierender Regeln
- quantitative Daten, wie Mengen und Bearbeitungs- oder Transportzeiten

Zur Erstellung der Supply Chain Modelle und deren Simulation gibt es vier alternative Implementierungsszenarien:

1. Das Simulationstool bietet eine eigene Modellierungsumgebung an und die im vorherigen Kapitel aufgestellten Anforderungen zur Modellierungssprache sind erfüllt. Dann kann diese Sprache direkt im Simulationstool als Basis für die Modellierungsmethode herangezogen werden. Beispiele für solche Tools sind AnyLogic (© The AnyLogic Company), Areena (© Rockwell Automation), Tecnomatix Plant Simulation (© Siemens Product Lifecycle Management Software Inc.), Simio (© Simio LLC).

2. Der Simulationsalgorithmus des Simulationstools basiert auf einer proprietären Modellierungssprache, die jedoch die Anforderungen nicht erfüllt. Das bedeutet, dass eine eigene domänenspezifische Methode erstellt werden muss und die damit erstellten Modelle vor der Simulation übersetzt und in das Tool übertragen werden müssen.

3. Das Simulationstool verfügt über keine graphische Modellierungskomponente, stellt aber eine Importschnittstelle zur Verfügung. Dann muss ein Standard zur Übertragung der Modellinformationen aus dem Metamodellierungstool in das Simulationstool ausgewählt und angewendet werden.

4. Das Metamodellierungstool bietet einen Simulationsalgorithmus an der direkt auf die erstellten Modelle angewendet werden kann.

Im nächsten Kapitel werden verschiedene bestehende Modellierungssprachen dahingehend untersucht, ob sie die hier aufgestellten Anforderungen erfüllen. Sollte diese Analyse ergeben, dass eine domänenspezifische Methode entwickelt werden muss, dann kommen die Implementierungsalternativen 2, 3 oder 4 infrage.

4.3.2 Transformation

Eine Transformation eines Modells in eine andere Notation, wie in Alternative 2 beschrieben, ist immer dann notwendig, wenn Funktionalität, die auf das Modell angewendet werden soll, nur für eine spezielle Notation entwickelt wurde. Wie aus Abbildung 41 ersichtlich, wird das Ausgangssimulationsmodell SM_1 durch die Transformation in ein Zielmodell SM_2 überführt.

Abbildung 41: Transformation von Simulationsmodellen

Sehr viele Auswertungsverfahren wurden beispielsweise für Petri Netze entwickelt, deshalb wird diese Sprache in vielen Literaturbeiträgen als Zielsprache einer Transformation verwendet (Hinz et al., 2005), (Lohmann et al., 2009). Vorteil dieses Vorgehens ist, dass nur ein Algorithmus für eine Sprache, nämlich die Zielsprache entwickelt und implementiert werden muss und es den Metamodellierern erlaubt, die Ausgangssprache domänenspezifisch anzupassen ohne den oder die Auswertungsalgorithmen anzufassen. Adaptiert werden muss somit nur der Transformationsalgorithmus, was in simplen Fällen der Anpassung einer Abbildungsfunktion mittels einer Mapping-Datei entspricht.

Offensichtlicher Nachteil jeder Modelltransformation ist, dass sich das, vom Modellersteller, meistens in Zusammenarbeit mit dem Fachbereich, erstelltem Ausgangsmodell, von dem Zielmodell unterscheidet. Gegeben falls unterstützt die Zielmodellsprache und der Transformationsalgorithmus nicht alle Konzepte der Ausgangssprache, was zunächst zu Abbildungsdefiziten im Zielmodell führt und im Weiteren verfälschte Simulationsergebnisse produzieren kann. Ein weiteres Problem der Simulation auf speziellen Simulationsmodellen ist, dass eine live-Animation ebenfalls auf dem Simulationsmodell abgebildet wird. Der Betrachter mag hierbei Probleme der Wiedererkennung und Akzeptanz haben, da sich das von ihm modellierte Modell vom animierten Modell unterscheidet.

Ähnliche Probleme bestehen bei der Transformation von Geschäftsprozessmodellen in ausführbare Workflowmodelle (Junginger, 2001), siehe Abbildung 42.

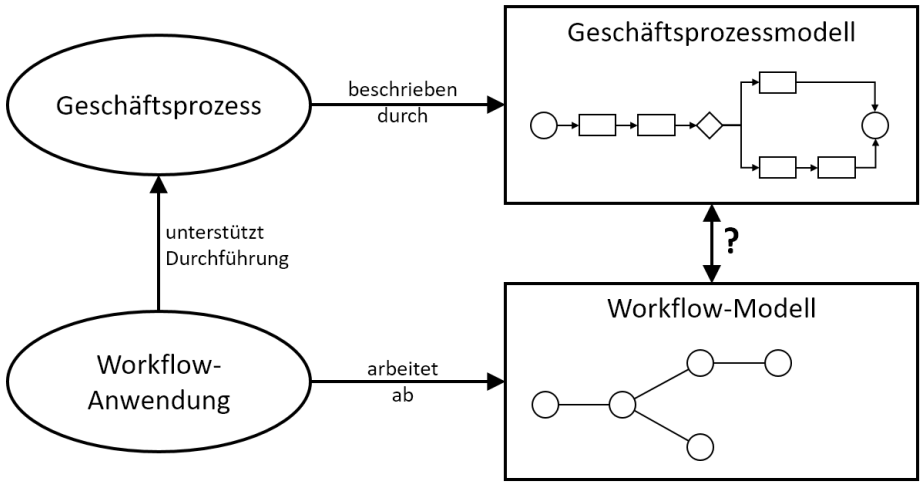


Abbildung 42: Zielsetzungen von Geschäftsprozess- und Workflow-Modellen (Junginger, 2001)

Es kann notwendig sein, mehrere Modelle zu einem Zielmodell zu transformieren oder sogar, je nach Ausgestaltung des Metamodells der Methode, Modelle unterschiedlicher Modelltypen in ein, für den Simulationsalgorithmus verständliches, Modell zu transformieren.

Prackwieser et al. (Prackwieser et al., 2014; Prackwieser et al., 2013) beschreiben in ihrer Arbeit einen hybriden Simulationsalgorithmus, der Geschäftsprozessmodelle unterschiedlicher Notationen simulieren kann und direkt in der Ausgangssprache animieren kann. Die Modelltransformation findet dabei rein auf Objektebene mittels einer semantischen Abbildung auf einen standardisierten Objekttyp statt. Dieser Algorithmus funktioniert allerdings nur für kontrollflussorientierte Modellierungssprachen, wie ADONIS BPMS, BPMN, Value Chains oder EPK. Nicht aber für gemischte Flussdarstellungen wie sie zur Abbildung von Material- und Informationsfluss notwendig sind.

4.3.1 Datenschnittstelle

Wird die Erstellung und Pflege der Modelle in einem anderen Werkzeug durchgeführt als die eigentliche Simulation, so ist eine Schnittstelle zwischen den beiden Systemen vorzusehen. Werkzeugschnittstellen können auf unterschiedliche Arten klassifiziert werden. Hier wird die von Junginger (Junginger, 2001, S. 250f) vorgeschlagene Klassifikation verwendet. Wie aus Abbildung 43 ersichtlich, unterscheidet er zwischen Richtung, Art und Technischer Realisierung. Die Art der Schnittstelle beschreibt, ob eine, wie oben beschriebene Modelltransformation stattfinden muss, eine Integration der Werkzeuge mittels Referenzierung auf jeweils im anderen Werkzeug abgelegte Modellinhalte erfolgt oder ob die Modelle überhaupt in einem gemeinsam genutzten Repository liegen. Ist die Schnittstelle bidirektional, um beispielsweise die Modelle mit Simulationsresultate anzureichern, so werden besondere Anforderungen an die Güte der Transformation gestellt. Die Transformation eines Modells von der domänenspezifischen Sprache in die Simulationssprache und nach Durchführung der Simulation wieder zurück in die domänenspezifische Sprache muss ein Modell ergeben, das dem ursprünglichen Ausgangsmodell entspricht.

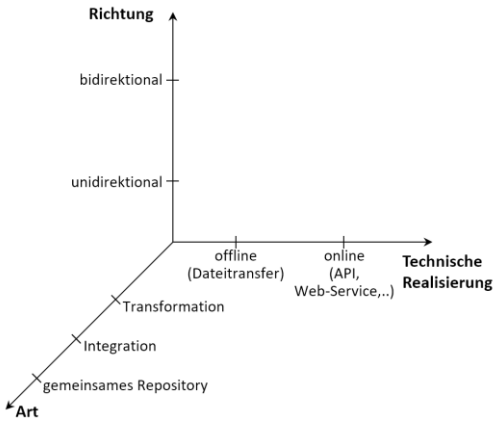


Abbildung 43: Beispiel zur Klassifikation von Werkzeugschnittstellen (Junginger, 2001)

Ob die technische Realisierung der Schnittstelle zwischen Modellierungstool und Simulationstool mittels einem manuell durchgeführten, file-basierten Datentransfer oder einer online Schnittstelle, wie beispielsweise einem Web-Service, siehe Abbildung 44 (www.adoxx.org), erfolgt, ist vom Resultat her gesehen unerheblich, mag aber Einfluss auf die Benutzerfreundlichkeit der Lösung haben.

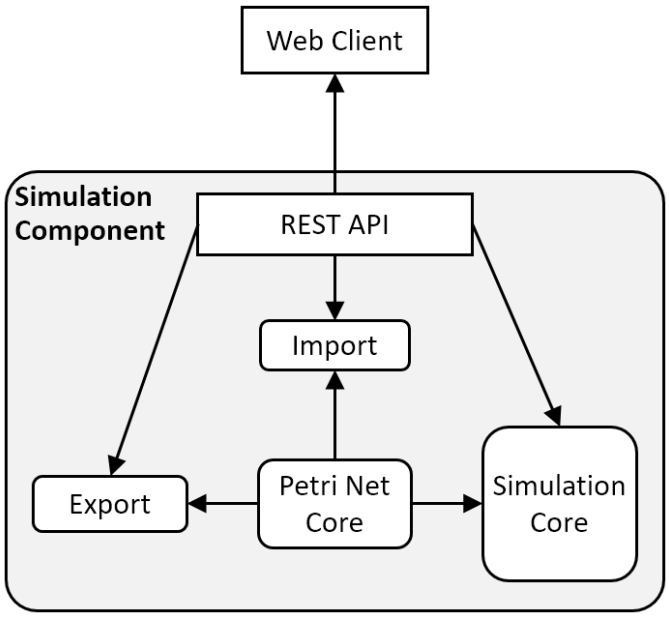


Abbildung 44: Ausschnitt aus ADOxxWEB Simulation (www.adoxx.org)

Um möglichst flexibel in der Verwendung von Simulationsalgorithmen zu bleiben, ist die Verwendung standardisierter Datenaustauschformate von Vorteil, die auf der eXtended Markup Language (XML) basieren. Bergmann (Bergmann, 2014) beschreibt in seiner Arbeit einige dieser Standards im Detail, speziell das Core Manufacturing Simulation Data (CMSD) Information Model das er zur Generierung von Simulationsmodellen basierend auf Produktionsdaten nutzt. Weitere Standards sind Simulation Data Exchange (SDX), Standard for The Exchange of Product model data (STEP) und Specification and Description Language (SDL).

4.4 Rule-Engine

Das Verhalten der Objekte der Supply Chain wird mittels Regelsets definiert. Diese Anweisungen werden je nach Realisierung entweder vom Modellierer in Attribute der betreffenden Objekte eingetragen oder als Referenz vom Objekt zu einer, in einem Regel-Repository definierten Regel gesetzt. Während der Simulation liest der Simulationsalgorithmus diese Regeln aus und leitet sie an die Regelmaschine weiter. Beispiele für einsetzbare Business Rule Management Systems (BRMS) sind Drools[57] als Teil der KIE (Knowledge Is Everything)-Familie, Red Hat JBoss BRMS[58] OpenL Tablets[59], OpenRules[60], IBM ILog®[61], FICO® Blaze Advisor®[62]. Vasilecas et al. (Vasilecas et al., 2016) bieten in ihrem Paper einen Überblick über einige dieser Tools und untersuchen ihre Eignung um die Dynamik von Geschäftsprozessen auszudrücken.

Es ist zu erwarten, dass sich die für die Implementierung gewählte Syntax der Regelsprache mit den Syntaxanforderungen der eingesetzten Rules Engine deckt. Somit entfällt weiterer Transformations- oder Übersetzungsaufwand zwischen Simulationsalgorithmus und Regelauswertung.

Neben den auszuwertenden Regeln benötigt die Rule-Engine als weiteren Input eine Fact Base, also eine oder mehrere Datenbanken in welchen zumindest folgende Informationen enthalten sein müssen:

- Abbildung der statischen Modellen, insbesondere Relationen zwischen den statischen Objekten
- Lokation und Status der dynamischen Objekte
- Aktuelle Simulationszeit
- Historie der bisherigen Simulationsresultate

Die Rule Engine kann die notwendigen Informationen zum ersten Aufzählungspunkt aus dem Modell-Repository des Modellierungstools auslesen, die restlichen Daten werden vom Simulationsalgorithmus erzeugt und in eine Datenbank übertragen. Dies muss unverzüglich erfolgen, dass manche Regeln den letztgültigen Status eines Objekts als Input benötigen.

Almeder et al. (Almeder et al., 2009) verwenden beispielsweise eine Microsoft® Access Datenbank, um die Modell- und Simulationsinformationen zu speichern, die vom

[57] http://www.drools.org Zugriff am 19.02.2017

[58] https://developers.redhat.com/products/brms/overview Zugriff am 19.02.2017

[59] http://openl-tablets.org Zugriff am 19.02.2017

[60] http://openrules.com Zugriff am 19.02.2017

[61] https://www-01.ibm.com/software/info/ilog Zugriff am 19.02.2017

[62] http://www.fico.com/en/products/fico-blaze-advisor-decision-rules-management-system Zugriff am 19.02.2017

Simulationstool und einem Optimierungstool verwendet werden (Almeder und Preusser), (Almeder und Preusser, 2007b).

4.5 Animations- und Reportkomponente

Der Modellierer bildet das statische Modell mittels einer graphischen Modellierungssprache ab und ergänzt es um Regeln, welche die dynamischen Abläufe im Modell beschreiben. Die Simulation ermittelt die Zustandsänderungen im Zeitablauf und aktualisiert die Eigenschaften betroffener Objekte. Dies kann unter anderem zu Bewegung von dynamischen Objekten, wie Teilen oder Nachrichten, oder zur Belegung von statischen Objekten, wie Produktionsprozessen, führen. Um diese Bewegungen sichtbar zu machen, wird eine Animationskomponente benötigt. Wie in den Anforderungen beschrieben, enthält das konzeptionelle Modell der Methode zwei Animationsarten. Eine Live-Animation, die während der Simulation die wichtigsten Informationen zum Status des Simulationsablaufs direkt im graphischen Modell darstellt und eine offline Animation, die ein Durchsteppen des Modells durch eine, vom User gewählte Betrachtungsperiode ermöglicht.

Die Reportkomponente stellt zur besseren Visualisierung der Simulationsergebnisse die zentralen Analyseergebnisse direkt im graphischen Modell, sozusagen am Ort ihrer Entstehung, dar. Zusätzlich kann die Reportkomponente tabellarischen Reports generieren. Dies erfolgt durch Zugriff auf das Modellrepository und die Simulationsdatenbank (Fact Base)

Die Reportkomponente liefert unter Nutzung der Visualisierungsfunktion der Animationskomponente beliebig viele exemplarische Supply Chain Modelle, die den Zustand des Systems zu einem Zeitpunkt beziehungsweise einer zuvor definierten Zeitdauer darstellen. Die zugrundeliegende Idee ist, dass eine sequenzielle Abfolge dieser exemplarischen zeitpunktbezogenen Modelle, wie eine Bildergeschichte oder ein Comicstrip, die dynamischen Zustandsänderungen des Modells darstellt. Um dies zu erreichen, muss die Ablaufreihenfolge der Zustandsänderungen, ausgedrückt in Eigenschaftsänderung der dynamischen Objekte, klar erkennbar sein. Ein in der exemplarischen Geschäftsprozessmodellierung mittels Kooperationsbildern bewährter Ansatz ist die Nummerierung der einzelnen Interaktionen zwischen Akteuren des Prozesses. Die resultierenden Modelle, werden als *Zustandsfolgediagramme* bezeichnet.

5. Metamodell

5.1 Untersuchung existierender Modellierungssprachen

In den vorherigen Kapiteln wurden Anforderungen an die Modellierungsmethode abgeleitet und beschrieben. Eine grundlegende Festlegung war dabei, dass die, in dieser Arbeit entwickelte Methode auf einer graphischen Modellierungsmethode für Supply Chains basieren soll.

Um zu entscheiden, ob dafür eine bereits existierende graphische Modellierungsmethode verwendet werden oder eine bestehende Notation erweitert werden kann, wurden die gesammelten Anforderungen dahingehend bewertet, ob sie als Kriterium für diese Entscheidung in Frage kommen. Der Autor dieser Arbeit, der sich seit über 20 Jahren mit Modellierungssprachen befasst und bereits selber etliche entwickelt hat, identifizierte eine zentrale Anforderung, deren Erfüllung für die Methode unerlässlich ist, die aber zugleich einen Großteil der bekannten Methoden ausschließt. Dabei handelt es sich um Anforderung 6 nach der Abbildungsmöglichkeit individueller dynamischer Objekte, also den, in der Maintenance Supply Chain bewegten Teile und Informationen (Prackwieser, 2013). In Ergänzung zur statischen Abbildung dienen diese dynamischen Objekte dazu, den aktuellen Modellzustand darzustellen. In die Untersuchung miteinbezogenen wurden kontrollflussorientierte Geschäftsprozess-Modellierungssprachen und Notationen, die im Supply Chain- oder Produktions-Umfeld bereits Verwendung finden, wie beispielsweise das Wertstrom-Mapping (Erlach, 2010) und die Prozess-Orientierte Analyse (Meyer et al., 2005), siehe Tabelle 5. Viele existierende Geschäftsprozesssprachen bieten zwar Konzepte zur Modellierung von Informations- und Materialflüssen, diese werden aber meist nur als Annotationen zur detaillierteren Beschreibung des maßgeblichen Kontrollflusses betrachtet. In BPMN wird das Konzept des steuernden Informationsflusses mit Message Flows methodisch umgesetzt. Hierbei wird aber der Kontrollfluss des Empfängerprozesses und nicht der für diese Arbeit benötigte Materialfluss beeinflusst. Eine Analyse durchgeführt im Projekt ProcessLog kam zu ähnlichen Ergebnissen (Schneider O, 2011), hier wurde eine Methode entwickelt, welche Material- und Informationsflüsse zusammen mit den eingesetzten Ressourcen abbilden kann. Das zur Synchronisation der Maintenance Supply Chain erforderliche Konzept der dynamischen Objekte, wie Teile und Nachrichten, ist aber nicht Bestandteil der Methode. Informations- und Materialfluss sind zentrale Modellierungskonzepte der Methode Wertstromdesign (Rother und Shook, 2015), aber auch hier haben steuernde Informationen keinen Einfluss auf das Verhalten des Modells, sondern werden nur zur Dokumentation dargestellt.

Tabelle 5: Untersuchte Modellierungssprachen

Modellierungssprache	Abbildung transienter Objekte	Kommentar / Literaturverweis
System Dynamics	-	
Markovketten	-	
Warteschlangennetzwerke	+	Es können unterschiedliche Auftragsklassen abgebildet werden
Unified Modellig Language (UML)	-	
Kontrollflussorientierte Geschäftsprozessmodellierungssprachen, wie zum Beispiel BPMN, ADONIS BPMS, EPK	-	
SADT (IDEF 0)	-	(Arnold und Furmans, 2009, S. 215)
Prozess-Orientierten Analyse (POA)	-	(Meyer et al., 2005)
Höhere Petri Netze, wie Gefärbte Petri Netze, Attributierte Petri Netze FUNSOFT Netze, XML-/XSLT-Netze	+	(Arnold und Furmans, 2009, S. 216)
Wertstrom-Mapping	-	(Erlach, 2010), (Klevers, 2007)
Production Authorization Card (PAC) System	-	(Schneider et al., 2005)
Processlog	-	(Günthner und Schneider, 2011), (Schneider O, 2011)
ProC/B	-	(Bause et al., 2002), (Arns et al., 2002)
SCOR	-	siehe Kapitel 2.3.8

Nur Warteschlangenmodelle und gefärbte Petri Netze erfüllen das zuvor beschriebene Entscheidungskriterium. Modelle beider Methoden lassen sich analytisch auswerten und es existieren bereits eine Vielzahl unterschiedlicher Auswertungsalgorithmen (Prackwieser, 2014)). In Warteschlangenmodellen erfolgt die Auftragssteuerung allerdings ausschließlich aufgrund stochastischer Eingangsparameter und ist somit zur Abbildung eines Steuerungsprogramms, wie in Anforderung 9 gefordert, ungeeignet.

Die Marken eines gefärbten Petri Netzes eignen sich sehr gut den jeweiligen Systemzustand modellhaft darzustellen. Ein Nachteil der Modellierungsmethode ist jedoch, dass größere Petri-Netze leicht unübersichtlich werden können. Dies ist vor allem dadurch gegeben, dass durch die Beschränkung der Notation auf zwei Arten von Knoten, den Stellen und Transitionen, für die Abbildung der selben fachlichen Inhalte eine höhere Anzahl von Objekten benötigt wird, als zum Beispiel bei Modellierungssprachen wie BPMN, die viele, jeweils semantisch spezifische Objekttypen aufweisen. Diese zunächst subjektive Aussage kann durch eine spezifische Auswertung einer Arbeit von Dijkman et al. (Dijkman et al., 2008) objektiviert werden. Sie beschreiben darin eine automatische Transformation von BPMN-Modellen in Petri Netze und zeigen die resultierende Modellgröße und die zur Transformation benötigte Rechenzeit. In Tabelle 6 wurde der mengenrelevante Ausschnitt ihrer Ergebnisse aufgelistet. Während zur Modellierung der 13 Petri Netz-Modelle 546 Objekte erforderlich sind, so werden in BPMN nur 216 Objekte, also ca. 40%, benötigt, um denselben fachlichen Inhalt abzubilden. Damit erfüllen Petri Netze zur Abbildung von Supply Chains nicht die, mit Anforderung 4 aufgestellte, Anforderung nach eine einfach verständlichen und ausdrucksstarken Notation.

Tabelle 6: Vergleich der Modellgrößen von BPMN Modell und aus einer Transformation resultierendem Petri Netz; Rohdaten übernommen aus (Dijkman et al., 2008)

Modell	BPMN-Modell								Petri Netz		
Nummer	Task	Event	XOR	AND	Subprocess	Message	Exception	**Summe**	Stelle	Transition	**Summe**
1	11	2	9	2				24	31	34	65
2	7	4	4	4				19	23	21	44
3	9	8	3		2		2	24	35	39	74
4	4	2	2					8	10	10	20
5	3	2	2	2				9	12	11	23
6	4	8	4			4		20	24	20	44
7	5	12	4			5		26	31	25	56
8	2	2	4					8	11	12	23
9	5	2	2					9	11	11	22
10	5	2		2				9	11	9	20
11	6	4	2	2				14	19	16	35
12	6	4	3		2		1	16	20	19	39
13	12	4	10	2	1		1	30	38	43	81
Summe Objekte								**216**			**546**

Die grundlegende Anforderung nach der Abbildung von temporären Objekten wird somit von keiner Methode befriedigend erfüllt. Deshalb wurde auf eine weitergehende Untersuchung, ob eine Erweiterung einer bestehenden Methode durch domänenspezifische Objekttypen möglich ist, verzichtet.

Konsequenz dieser Untersuchung ist, dass zur Umsetzung der Methode SIMchronization eine neue domänenspezifische Modellierungsmethode entwickelt werden muss.

5.2 Entwicklung einer domänenspezifischen Methode

Nachdem die Anforderungen an die Modellierungsmethode formuliert sind und erkannt wurde, dass keine bestehende Modellierungssprache die gestellten Anforderungen erfüllt, wird in diesem Kapitel die Konzeption des Metamodells der Sprache beschrieben.

Das Design der Modellierungssprache wurde von verschiedenen Inputs und Einflussfaktoren geprägt:

- der Struktur des SCOR-Modells (beschrieben in Kapitel 2.3.8)
- Supply Chain spezifischen Ontologien
- Gesprächen mit wissenschaftlichen Kollegen und Praktikern
- der Erfahrung des Autors dieser Arbeit, der sich seit über 20 Jahren mit Modellierung und dem Engineering von Modellierungssprachen befasst.

Eine zentrale Designentscheidung für Modellierungssprache ist die, der zur Verfügung gestellten Sichten auf den Modellinhalt. Eine Sicht, in dieser Arbeit als Modelltyp bezeichnet, bündelt verschiedene Objekttypen (Klassen). Konzepte die einer Sicht zugeordnet werden müssen, sind Planung, Bearbeitung, Ressourcen (Mitarbeiter, Werkzeuge, IT-Systeme), Materialfluss, Informationsfluss und Teile.

Anforderung 4 gibt vor, dass den Nutzern der Methode eine intuitive, einfach verständliche Notation zur Verfügung gestellt wird. Es liegt daher nahe, die Strukturierung der Sichten an etablierte, allgemein bekannte Methoden anzulehnen. Die Unterteilung der organisationsrelevanten Dokumentation in eine aufbauorganisatorische- und eine ablauforganisatorische Sicht ist in vielen Unternehmen Standard[63]. Diese Strukturierung findet sich auch in vielen Modellierungssprachen wieder. So bietet ADONIS BPMS unter anderem Geschäftsprozess- und Arbeitsumgebungsmodelle zur Beschreibung der Organisation des Unternehmens an, IT-Systemmodelle zur Abbildung der

[63] Wenngleich bestimmte Methoden die Ablauforganisation mit Informationen über verantwortliche Organisationseinheiten anreichern, wie zum Beispiel mittels swim lanes.

Informationstechnologie und Dokumentenmodelle, um Dokumente und ihre Struktur modellieren zu können. Die Objekte des IT-Systemmodells, Dokumentenmodells und Arbeitsumgebungsmodells können dazu verwendet werden Geschäftsprozesse detaillierter zu beschreiben. Dokumente können als In- und Outputs zu Aktivitäten annotiert werden und somit implizit einen Informationsfluss bilden, der in ADONIS BPMS nicht explizit modelliert wird. Abbildung 45, zeigt das zugrundeliegende Konzept, in dieser frühen Version wurden Dokumente noch als Artefakte der Informationstechnologie angesehen.

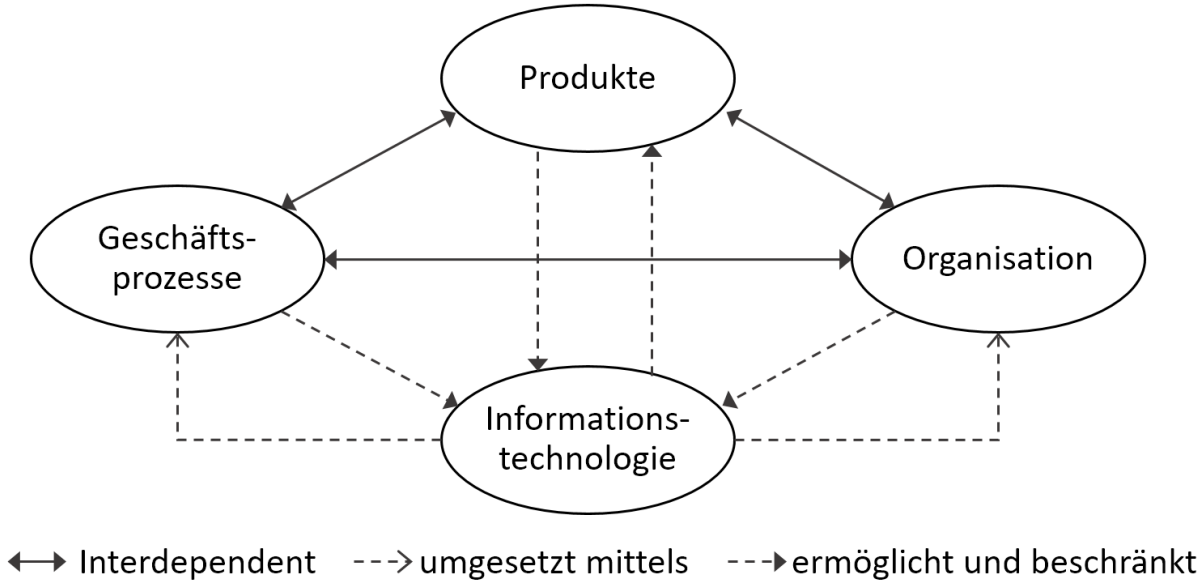


Abbildung 45: Kernelemente eines Unternehmens (Junginger et al., 2000)

Auch einige Ontologien, die das Supply Chain Management beschreiben, verwenden diese Strukturierung. So besteht die Onto-SCM (Ye et al., 2008b), die in Kapitel 5.4.2 und 5.5.1 detaillierter beschrieben wird, aus fünf Modulen, siehe Abbildung 46. Die Basismodule SC-Structure, SC-Activity, SC-Resource und SC-Item und das darauf aufbauende Modul SC-Management. SC-Activity entspricht dabei einer Prozesssicht, SC-Resource der Ressourcensicht und SC-Item der Teilesicht. SC-Structure definiert Basiskonzepte wie Strategie oder Ziele und ist wie das SC-Management Modul für die, in dieser Arbeit entwickelte Modellierungssprache nicht relevant. Die SC-Item Sicht ist hingegen sehr relevant, sie dient dazu Teile und Materialien näher zu beschreiben und in Beziehung zueinander zu bringen. Diese Beschreibungen sind in Form von Stücklisten und Bauteillisten ein, in Produktion und Logistik häufig verwendetes Konzept.

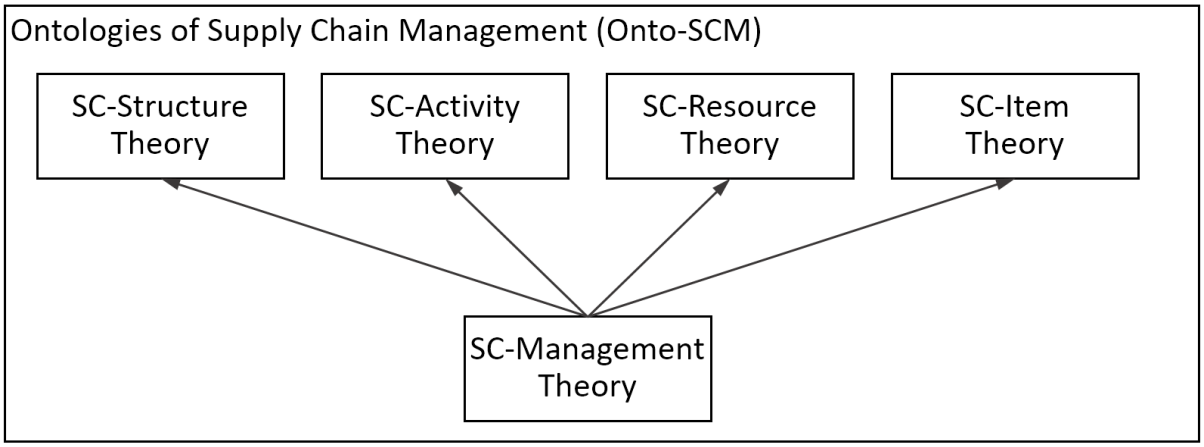


Abbildung 46: Struktur von Onto-SCM (Ye et al., 2008b)

Die vier zentralen Modelltypen zur Beschreibung der Maintenance Supply Chain sind somit:

Supply Chain Network Model[64], es bündelt alle Objekttypen die zur Abbildung der Material- und Informationsflüsse notwendig sind.

Resource Model, es enthält alle Objekttypen mit denen Ressourcen modelliert werden können, die im Supply Chain Network Modell genutzt werden.

Item Model, stellt Objekttypen zur Verfügung, um die dynamischen Objekte des Materialflusses (Teile) und deren Struktur zu definieren.

Information Model, umfasst Objekttypen, die zur Definition von Informationsobjekten notwendig sind.

Im Folgenden werden die einzelnen Komponenten des Metamodells im Detail erläutert. das aus dem Methodenengineering entstandene Metamodell ist in Abbildung 47 dargestellt. Es stellt eine Weiterentwicklung des von Prackwieser (Prackwieser, 2013) präsentierten Modells.

[64] Die Modellierungssprache wird in englischer Sprache implementiert, um sie einer breiteren wissenschaftlichen Community zur Verfügung stellen zu können.

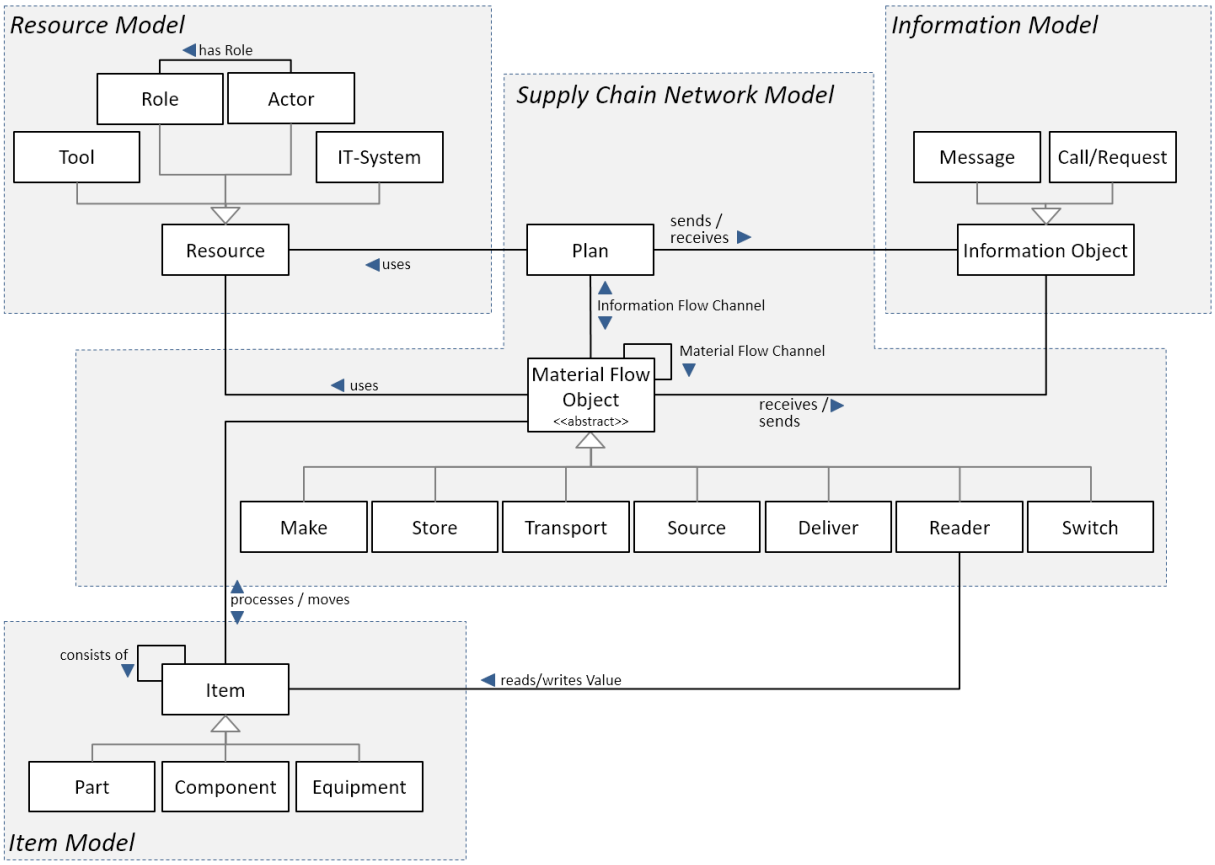


Abbildung 47: Metamodell der Methode SIMchronization

5.3 Modelltyp „Supply Chain Network Model“

Wie aus dem Metamodell in Abbildung 47 ersichtlich, fasst der Modelltyp „Supply Chain Network Model“ alle statischen Objekttypen zusammen, die zur Abbildung des Material- und Informationsflusses notwendig sind.

Das in Kapitel 2.3.8 beschriebene SCOR Modell erfüllt die Anforderung 4 am besten, wonach eine bekannte und domänenspezifischen Prozessstruktur und Terminologie verwendet werden soll. Demgemäß sind die in SCOR verwendeten Prozesse Plan, Make, Source, Deliver, Return und Enable primäre Kandidaten, um in das Metamodell mitaufgenommen zu werden. Folgende SCOR-Prozesse werden als Objekttypen in die Modellierungssprache übernommen:

Plan: um zentralisierten Steuerungskomponenten der Supply Chain abzubilden, die über den Informationsfluss andere Prozessobjekte steuern können.

Make: um Prozesse abzubilden, die dynamische Objekte, wie Teile, bearbeiten können.

Source: um Startpunkte des Materialflusses definieren zu können, über die, neue, dynamische Objekte in den Materialfluss eintreten.

Deliver: um Endpunkte des Materialflusses definieren zu können, an denen dynamische Objekte aus dem Materialfluss austreten.

Return: bündelt verschiedene Prozesse, die Teile, wie Verpackungen oder defekte Produkte, zurück zum Fabrikanten liefern. Da der Return-Prozess konzeptionell aus den anderen Prozessen zusammengestellt werden kann, wird er nicht als separater Objekttyp in das Metamodell mitaufgenommen.

Enable: stellt Grundlagen des Supply Chain Managements den anderen Prozessen zur Verfügung und ist nicht direkt in den Informations- oder Materialfluss miteingebunden. Deshalb wird der Enable-Prozess ebenfalls nicht mit in die Methode übernommen.

Nicht im SCOR-Modell enthalten, aber notwendig zur Erfüllung der gestellten Anforderungen sind die folgenden Objekttypen:

- Um die Zeit zwischen der Bereitstellung eines Teils und der Verwendung des Teils in der nächsten Stufe der Wertschöpfungskette überbrücken zu können, wird ein Objekttyp benötigt der es erlaubt, Teile zu lagern und diese Lagerzeiten und –mengen in der Analyse nachzuvollziehen. Es wird deshalb ein **Store**-Objekttyp der Methode hinzugefügt.
- Neben der Überbrückung von Zeit ist die Überbrückung der Lokation eine wichtige Funktion der Logistik, dies erfordert ein Modellierungsobjekt **Transport**.
- Um eine Verbindung zwischen zwei Objekten zu definieren, über die Information oder Material fließen kann, werden die Relationstypen „**Information flow channel**“ und „**Material flow channel**“ benötigt.
- Potentielle Verzweigungen im Materialfluss müssen definierbar sein, dafür wird der Objekttyp **Switch** eingeführt.
- Um Information auf Teilen (mittels RFID, Smarttag, Barcode…) auszulesen und gegeben falls aufzubringen wird ein Objekttyp **Reader** vorgesehen.

Im Folgenden werden die einzelnen Objekttypen, oder Modellierungsklassen näher beschrieben. Um die Methode einer breiteren Community zum Testen, Anwenden und Weiterentwickeln zur Verfügung stellen zu können, wurde die Notation in englischer Sprache entwickelt und implementiert. Auf Attribute wird nur insofern eingegangen, als dass sie direkt in Zusammenhang mit der Anwendung der Methode SIMchronization beziehungsweise der Dynamisierung des Modells stehen. Attribute, die beispielsweise zur Hinterlegung von Werten für das Supply Chain Risk Management, das Qualitäts- oder Business Continuity Management erstellt wurden, werden hier nicht besprochen.

Eine formale Beschreibung der Klassen und Attribute, inklusive des Attributtyps, findet sich in Kapitel 6 „Beschreibung des Metamodells in FDMM“.

Jeder statische Objekttyp verfügt über ein Attribut „Name“ und ein Attribut „Description“. In Name wir das Objekt fachlich benannt und identifiziert und in Description kann der Prozess näher verbal erläutert werden und auf Details eingegangen werden, die für eine graphische Modellierung nicht geeignet sind.

5.3.1 Modellierungsklasse „Make“

Objekte, die von der Klasse Make abgeleitet sind, beschreiben die Tätigkeiten im Materialfluss, die mit der Gewinnung, Bearbeitung und Verarbeitung von stofflichen Gütern befasst sind (VDI, 1970). Darunter fallen zum Beispiel Produktionsschritte, Inspektionen, Wartungs- und Instandsetzungstätigkeiten. Werden Teile nur einer geringen Ortsänderung unterzogen, wie beispielsweise beim Ver- oder Entladevorgang mit einem Stapler, so wird dieser Vorgang mittels Make-Objekten abgebildet[65]. Verpackungs- und Entpackungsvorgänge sind ebenfalls Zustandsänderungen von Teilen, die mit einem Make-Objekt modelliert werden. Zur Erklärung der syntaktischen Anwendung und semantischen Aussage eines Make-Objekts wird zunächst auf den Standardfall eingegangen. Ein Produktionsschritt benötigt einen definierten Input zur Erzeugung eines bestimmten Outputs.

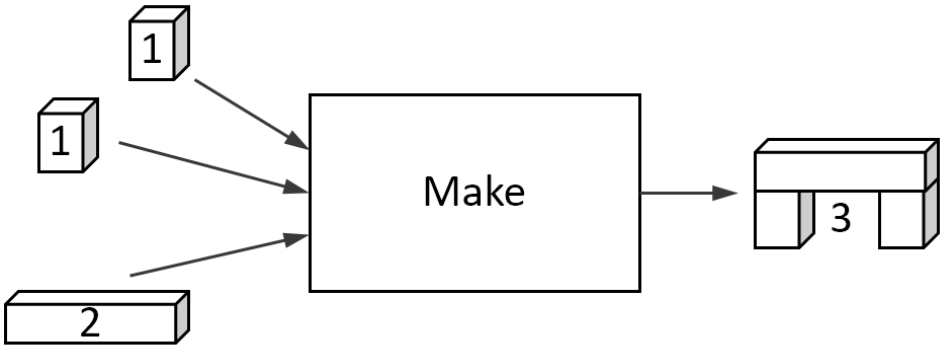


Abbildung 48: Beispiel für ein- und ausgehende Materialflüsse in ein Make-Objekt

Abbildung 48 zeigt eine leicht verständliche Darstellung des Materialflusses bei der Produktion eines Teils 3. Es werden zur Erzeugung eines Teils vom Typ 3, genau 2 Teile vom Typ 1 und ein Teil vom Typ 2 benötigt und verbraucht.

In der Modellierungssprache SIMchronization wird jedoch zunächst nicht der tatsächliche Materialfluss abgebildet, sondern im statischen Supply Chain Network Modell, die Kanäle modelliert, über die Material in den Make-Prozess gelangen kann, beziehungsweise der Kanal modelliert, aus dem das produzierte Stück austritt. Ob über einen Kanal tatsächlich Material fließt, hängt vom Zustand und den Anforderungen des Systems zum jeweiligen Zeitpunkt

[65] Ist die zu überbrückende Entfernung größer, so wird für Ortsänderungen ein Objekt der Klasse Transport verwendet

während der Simulation ab. Umgelegt auf ein Modell bedeutet dies, dass ein Make-Objekt, in das fünf Material flow channels eingehen, nicht 5 Rohmaterialien zur Produktion benötigen muss.

Dieses auf den ersten Blick vielleicht umständliche Modellierungsverfahren, hat den Vorteil, dass die statische Struktur, also das graphische Modell, des Systems bei der Simulation verschiedener Szenarien weitgehend stabil bleiben kann. Dem liegt die Überlegung zu Grunde, dass bei der Verbesserung der Synchronisation zunächst gar nicht das Fabriklayout und dessen Materialflussaggregate, wie Förderbänder und Laufkräne, verändert werden müssen, sondern Verbesserungen bereits durch eine Veränderung oder Beschleunigung des Informationsaustausches erfolgen können. Dies hat auf die Beschreibung des SCN-Modells die Auswirkung, dass das graphische Modell nicht immer verändert werden muss, sondern viele Veränderungen durch ein Anpassen das Verhalten einzelner SC-Objekte bewirkt werden können.

Ein plastisches Beispiel ist ein Make-Prozess einer Instandhaltungstätigkeit mit zwei eingehenden Materialflusskanälen über die Ersatzteile desselben Typs geliefert werden können. Ein Kanal kommt von einem günstigen Inhouse-Lager, der zweite Kanal von einem teureren Lager eines externen Dienstleisters. Ist während der simulierten Periode das hausinterne Lager immer gefüllt, so wird der Make-Prozess von diesem Lager bestückt. Ist der Lagerstand aber nicht ausreichend und die Instandhaltungsmaßnahme muss dringend erfolgen, so kann auf das externe Lager zugegriffen werden.

Um einen Make-Prozess für eine dynamische Auswertung zu definieren, müssen folgende Informationen angegeben werden.

- Anzahl der Teile je Teiletyp die für einen Produktionsprozess als Input erforderlich sind
- Anzahl der Teile je Teiletyp die sich aus einen Produktionsprozess ergeben
- Bearbeitungszeit
- Anzahl der Prozessdurchführungen

5.3.1.1 Attribut „Production program“

Ein Make-Prozessschritt benötigt als Input eine Kombination verschiedener Teiletypen in jeweils definierter Menge. Nach einer Bearbeitungszeit, liefert der Produktionsschritt als Output eine bestimmte Anzahl von Teilen pro Teiletyp. In einigen Modellierungssprachen, wie beispielsweise in gewichteten Petri Netzen, wird die Angabe der benötigten Inputmengen und der gelieferten Output-Mengen direkt an die ein- und ausgehenden Kanten des Materialflusses geschrieben. Wenngleich dies Vorteile für die Lesbarkeit des Modells bedeutet, erschwert es die flexible Anpassung von Produktionsverfahren während der

Simulation. Eine alternative Belieferung eines Make-Prozesses aus zwei Lagern ist mittels dieser statischen Beschreibung der benötigten Mengen nicht ohne Modellierung weiterer SC-Objekte abbildbar. Ein weiterer Grund, der gegen die Hinterlegung der Teilemengen direkt an den eingehenden Kanten eines Make-Objekts spricht, ist das, der Methode SIMchronization eigene, Konzept des Material flow channels. Diese stellen nur die prinzipielle oder technische Möglichkeit eines Materialflusses zwischen zwei statischen Objekten dar. Ob es tatsächlich zu einer Bewegung über einen Kanal kommt, bestimmt das Verhalten oder der Status der beiden adjazenten SC-Objekte an den jeweiligen Endpunkten des Kanals während der Simulation.

Um die geforderte Flexibilität zu erreichen, werden benötigte In- und erzeugte Outputs direkt in dem Attribut „Produktionsprogramm“ des Make-Objekts definiert. Die Syntax zur Beschreibung eines Produktionsprogramms hängt von der Art der Tätigkeit ab. Tätigkeitstypen sind:

- Produktion oder Bearbeitung
- Verpacken
- Entpacken

Wobei der Unterschied hauptsächlich darin liegt, dass bei Produktionsprozessen, die verarbeiteten Teile zu einem späteren Zeitpunkt nicht mehr extrahiert werden können, während dies bei Verpackungsprozessen durch das Entpacken wieder möglich ist. Die drei Bearbeitungsweisen sind alternativ zu sehen, es ist also nicht möglich in einem Schritt eine Produktion und Verpackung vorzunehmen. Für diesen Fall sind zwei Make-Objekte erforderlich.

Die Syntax des Produktionsprogramms wird im Kapitel „Implementierung“ definiert.

Zur Produktion von Teilen eines bestimmten Typs werden folgende Inputs benötigt:

Verbrauchte Teile: Teile, die zur Produktion eines anderen Teils benötigt werden und im Produktionsvorgang verbraucht, unauflösbar verbaut, zerstört (zum Beispiel im Rahmen einer Prüfung) oder so verändert werden, dass sie ihren Typ ändern. Für diese dynamischen Objekte stellt das Make-Objekt einen Endpunkt im Materialfluss dar.

Erzeugte Teile: sind das Ergebnis eines Make-Prozesses. Für diese dynamischen Objekte stellt das Make-Objekt den Startpunkt im Materialfluss dar.

Bearbeitete Teile: sie werden bei der Produktion nicht verbraucht, sondern nur gebraucht oder leicht verändert. Beispiele könnten Werkzeuge oder Messinstrumente sein, deren logistischer Prozess relevant für die Betrachtung ist. Bearbeitete Teile sind auch Teile die vom Make-Prozess nur einer geringen Veränderung unterzogen wurden und somit keine Typänderung rechtfertigen. Make-Objekte die Inspektions- oder Wartungstätigkeiten (wie Ölen) abbilden, werden das instand zu haltende Teil

bearbeiten. Die Anzahl der in ein Make-Objekt eingehenden bearbeiteten Teile ist gleich zu der Anzahl ausgehender Teile.

Die Verpackung von Teilen erfordert mindestens zwei Inputs von Teilen mit unterschiedlichem Typ. Ein Teil muss ein Behälter oder die Verpackung sein und ein oder mehrere andere Teile das zu verpackende Gut. Dabei ist es nicht notwendig, dass in einem Make-Objekt sofort alle zu verpackenden Teile dem Behälter zugeführt werden, dies ist auch schrittweise über mehrere Make-Objekte hinweg möglich.

Der gefüllte Behälter und alle darin enthaltenen Teile werden im weiteren Verlauf des Materialflusses als ein Teil betrachtet. Ein physischer Zugriff auf die einzelnen verpackten Teile ist nur nach einem Entpackvorgang möglich. Sollten die Teile jedoch über RFID-Tags verfügen, können ihre Informationen im „Bulk“ ausgelesen werden (Reader-Funktionalität „Bulk-Read/Write“ (siehe Kapitel 5.3.7). Beim Entpacken einzelner Teile aus dem Behälter oder der Auflösung des gesamten Verbundes werden wieder alle Teile als individuelle Teile im Materialfluss betrachtet.

Allen drei Bearbeitungsweisen ist gemein, dass sie Zeit zur Bearbeitung benötigen. Diese wird ebenfalls im Produktionsprogramm angegeben. Sollte sich über den Zeitverlauf hinweg ein Parameter des Produktionsprogramms ändern, so muss im Attribut Produktionsprogramm ein entsprechender Regelausdruck definiert werden.

5.3.1.2 Attribut „Order“

Dieses Attribut enthält die Anzahl der, in einer Periode zu startenden Aufträge. Eine Auftragshöhe von“0“ bedeutet, dass das Make-Objekt in dieser Periode keine neuen Aufträge durchführt. Der Wert des Attributs ist typischerweise während einer Simulation nicht stabil, sondern kann von anderen Objekten mittels Nachrichten oder vom Objekt selbst, entsprechend einer Regel, verändert werden:

Ständige Produktion. Durch eintragen einer Konstanten in das Attribut

Produktion zu definierten Zeitpunkten. Durch Definition einer zeitpunktbezogenen Regel

Produktion in Abhängigkeit des Zustandes anderer SC-Objekte. In dem beispielsweise mittels eines Information flow channels eine Abfrage über den Lagerstand eines Store-Objekts gemacht wird.

Ist der resultierende Wert im Attribut „Order“ größer 1, so entspricht diese Zahl einem Multiplikator, der auf das Produktionsprogramm angewendet wird. Es wird somit ein, dem Multiplikator entsprechendes Vielfaches der Teile benötigt und auch produziert, die Bearbeitungszeit bleibt jedoch gleich.

Kann ein Auftrag nicht ausgeführt werden, beispielsweise weil der Prozess belegt ist oder die benötigten Input-Teile nicht zur Verfügung stehen, so verfällt der Auftrag. Er wird im Make-Objekt nicht für spätere Perioden gespeichert, könnte aber über Regeln, die auf die Auftragshistorie zugreifen, ermittelt werden.

5.3.1.3 Attribut „Resource“

Kann zur Referenzierung von einer oder mehrerer Ressourcen, wie Maschinen oder Personal, verwendet werden, die zur Durchführung des Make-Prozesses erforderlich sind. Da das primäre Ziel dieser Methode nicht eine Ressourceneinplanung ist, wird dieses Attribut nur zur Dokumentation der benötigten Ressourcen verwendet.

5.3.1.4 Attribut „Fixed costs per Order“

Kosten, die einmal bei der Erteilung eines Auftrags und dessen erfolgreichem Start anfallen.

5.3.1.5 Attribut „Execution time variable costs“

Kosten, pro Periode der Bearbeitung.

5.3.1.6 Attribut „Production history“

Dieses Attribut wird während einer Simulation vom Simulationsalgorithmus mit Werten gefüllt. Für jede Periode wird die Anzahl der angestoßenen, also beauftragten und begonnenen Aufträge in diesem Attribut erfasst. Andere Objekte können per Abruf auf dieses Attribut zugreifen und diese Information für Steuerungsaufgaben verwenden.

5.3.1.7 Attribut „Remaining production time“

Dieses Attribut wird während einer Simulation vom Simulationsalgorithmus gesetzt. Es wird in jeder Periode vom Simulationsalgorithmus, der Zeitraum berechnet, in dem das Make-Objekt noch vom aktuellen Auftrag belegt ist. Auf diese Information kann von anderen Objekten aus per Abruf zugegriffen werden und daraus Steuerungsentscheidungen getroffen werden.

5.3.1.8 Relationen von Make-Objekten

Objekte der Klasse Make können sowohl über Relationen der Klasse Material flow channel als auch über Objekte der Relationsklasse Information flow channel mit Objekten anderer Klassen

verbunden sein. Dabei gelten folgende Kardinalitäten für die Modellierung eines Make-Objekts:

Tabelle 7: Kardinalitäten eines Make-Objekts

K1	es hat 0...n eingehende Material flow channel
K2	es hat 0..1 ausgehende Material flow channel
K3	es hat 0..m eingehende Information flow channel
K4	es hat 0..m ausgehende Information flow channel
K5	eingehende Material flow channel gehen immer direkt von einem Objekt der Klasse Store oder Transport aus
K6	ein ausgehender Material flow channel führt zu einem Objekt der Klasse Store, Transport, Reader oder Switch

Ist kein eingehender, beziehungsweise ausgehender Material flow channel vorhanden, so handelt es sich um einen Sonderfall. Diese Option eröffnet die Möglichkeit das Make-Objekt auch zur Abbildung von rein manuellen Tätigkeiten, die keinen Materialfluss erfordern, zu verwendenden. Ein Output solcher Tätigkeiten wären dann beispielsweise generierte Daten oder Informationen.

Da nur maximal ein ausgehender Material flow channel erlaubt ist, muss für den Fall, dass aus dem Make-Objekt Teile unterschiedlichen Teiletyps resultieren, zwischen dem Make-Objekt und den nachfolgenden Store-Objekten ein Switch gesetzt werden.

5.3.1.9 Domänenspezifische Besonderheiten

Bei der Abbildung einer ganzheitlichen Maintenance-Supply Chain werden Objekte der Klasse Make in verschiedenen Kontexten verwendet:

- In der Beschaffungs-Supply Chain von Ersatzteilen
    - Qualitätssicherung von Ersatzteilen
    - Verladen- und Entladen
    - Verpacken- und Entpacken
    - Eigenproduktion von Ersatzteilen
- In der Wartung und Inspektion
    - Prüfvorgänge
    - Wartungstätigkeiten
- In der Instandsetzung
    - Reparatur von Anlagen

- Austausch von Teilen
- Reparatur von Ersatzteilen

Der Sonderfall, dass kein Vorgänger oder Nachfolger vorhanden ist, wird beispielsweise zur Abbildung von Inspektionstätigkeiten genutzt, die bei laufendem Betrieb der Maschine, direkt an deren Einsatzort stattfinden. Das zu inspizierende Teil bewegt sich somit nicht.

5.3.2 Modellierungsklasse „Store“

Ein Objekt der Klasse Store bildet die Zeitüberbrückung vom Zeitpunkt der Bereitstellung eines dynamischen Teils bis zum Zeitpunkt des Bedarfs ab. Es werden also Teile gelagert oder gepuffert um Bedarfsschwankungen im Produktionsprozess auszugleichen.

5.3.2.1 Attribut „Item type“

In einem Store können eine beliebige Anzahl von Teilen genau eines Teiletyps gelagert werden. Die Typisierung eines Stores definiert somit eindeutig den Teiletyp der Teile, die über die ein- und ausgehenden Material flow channels des Stores bewegt werden können.

5.3.2.2 Attribut „Condition adjustment“

Während der Lagerung können die Teile einer Zustandsänderung unterliegen. Das kann positive und meist gewünschte Auswirkungen, wie das Nachreifen und damit verbundene Wertsteigerung von Lebensmittel oder negative Auswirkungen auf die Qualität des gelagerten Teils haben.

Das Maß der Zustandsänderung für jedes gelagerte Teil in einem Store kann mittels einer Formel definiert werden, die als Regel formuliert wird. Im Attribut „Condition adjustment“ wird diese Regel hinterlegt. Da in einem Store nur Teile desselben Teiletyps gelagert werden, ist die zugrundeliegende Formel für alle Teile identisch.

Im Unterschied zu einem produzierenden Make-Prozess geht die Zustandsänderung aber nicht soweit, dass ein neuer Typ des Teils entsteht. Vielmehr bleibt der Typ des gelagerten Teils identisch, aber seine Qualität ändert sich. Um diese quantitative Kennzahl abzubilden, führt jedes Teil, das durch das Supply Cain Network Modell bewegt wird, ein internes Attribut mit, das im Fall der Zustandsänderung durch Lagerung, gemäß dem Ergebnis der Formel während der Simulation verändert wird. Ein Auslesen dieses Attribut ist durch eine Kombination von einem Make- und einem Reader-Prozess möglich.

5.3.2.3 Attribut „Condition history“

Dieses Attribut wird während einer Simulation vom Simulationsalgorithmus mit Werten gefüllt. In jeder Periode wird der aktuelle Wert für den Zustand des am längsten in dieses Lager eingelagerten Teils in dieses Attribut vom Simulationsalgorithmus eingetragen. Andere Objekte können per Abruf auf dieses Attribut zugreifen und diese Information für Steuerungsaufgaben verwenden.

5.3.2.4 Attribut „Stock level“

Aktuelle Anzahl von Teilen des in diesem Store-Objekt lagerbaren Teiletyps, die abrufbar im Lager liegen. Ein Teil, das in das Lager eingelagert wird, steht erst mit Beginn der neuen Periode wieder zum Abruf bereit. Der Lagerstand wird vom Simulationsalgorithmus ermittelt und gesetzt. Die Simulation aktualisiert dieses Attribut nach jedem Lagerabgang und am Ende einer Periode um alle Zugänge während dieser Periode zu berücksichtigen.

5.3.2.5 Attribut „Resource“

Kann zur Referenzierung eines physischen Lagerplatzes abgebildet als Objekt im Ressourcen Modell verwendet werden, dient ausschließlich der Dokumentation.

5.3.2.6 Attribut „Storage history“

Dieses Attribut wird während einer Simulation vom Simulationsalgorithmus mit Werten gefüllt. Am Ende jeder Periode wird der aktuelle Lagerstand in dieses Attribut eingetragen. Andere Objekte können per Abruf auf dieses Attribut zugreifen und diese Information für Steuerungsaufgaben verwenden.

5.3.2.7 Attribut „Items in stock“

Dieses Attribut wird während einer Simulation vom Simulationsalgorithmus mit Werten gefüllt. Das Attribut enthält alle Teile, die im Lager vorgehalten sind und zur Entnahme bereitstehen.

5.3.2.8 Attribut „Items waiting to be stored“

Dieses Attribut wird während einer Simulation vom Simulationsalgorithmus mit Werten gefüllt. Das Attribut enthält alle Teile, die gerade im Lager eingetroffen sind und erst in der nächsten Periode zur Entnahme bereitstehen.

5.3.2.9 Attribut „Handling cost per item“

Gesamtkosten, der Ein- und Auslagerungstätigkeit pro Teil

5.3.2.10 Attribut „Storage cost per period and item“

Kosten, die pro Periode und gelagertes Teil anfallen

5.3.2.11 Relationen von Store-Objekten

Objekte der Klasse Store können sowohl über Relationen der Klasse Material flow channel als auch über Objekte der Relationsklasse Information flow channel mit Objekten anderer Klassen verbunden sein. Dabei gelten folgende Kardinalitäten für die Modellierung eines Store-Objekts:

Tabelle 8: Kardinalitäten eines Store-Objekts

K1	es hat 0...n eingehende Material flow channel
K2	es hat 0.. n ausgehende Material flow channel
K3	es hat keine eingehenden Information flow channel
K4	es hat 0..m ausgehende Information flow channel
K5	ausgehende Material flow channel gehen immer direkt zu Objekten der Klasse Make oder Deliver
K6	eingehende Material flow channel kommen von einem Objekt der Klasse Make, Source, Reader oder Switch

Dass ein Store-Objekt keinen eingehenden Material flow channel hat, stellt einen Sonderfall dar, der zu modellieren nur sinnvoll ist, wenn das Store-Objekt vom Modellnutzer mit Teilen belegt wird, um einen bestimmten Anfangszustand der Simulation zu erzwingen. Dies kann sinnvoll sein, wenn das System sofort im eingeschwungenen Zustand simuliert werden soll, also alle Lager bereits wie im normalen operativen Betrieb gefüllt sind.

In Sonderfällen kann es des Weiteren sinnvoll sein, einen Materiafluss nicht mit einem Deliver-Objekt abzuschließen, sondern mit einem Store-Objekt. Beispielsweise, wenn nur ein Ausschnitt eines Prozesses dargestellt wird und fachlich ein Abschluss des betrachteten Materialflusses mit einem Deliver-Prozess keinen Sinn macht. Da Teile, die in ein Store-Objekt ohne ausgehenden Material flow channel bis zum Ende der Simulation im Store-Objekt

verbleiben, kann diese Modellierungsvariante dafür genutzt werden, die Entwicklung von Teilemengen plastisch darzustellen.

5.3.2.12 Domänenspezifische Besonderheiten

Zwischen den Objekten der Klasse Make, Source und Deliver muss für jeden eingehenden Material flow channel mindestens ein Store-Objekt modelliert werden. Deshalb haben diese Puffer- oder Lagerprozesse in der gesamten Maintenance Supply Chain unterschiedliche Aufgaben:

- In der Beschaffungs-Supply Chain von Ersatzteilen
    - Eingangslager
    - Puffer vor Qualitätssicherung
- In der Wartung und Inspektion
    - Ein Store im Instandhaltungsprozess dient dazu, das Servicepersonal im Bedarfsfall mit Ersatzteilen zu versorgen. (Kennedy et al., 2002)
- In der Instandsetzung
    - Lager für ausgewechselte Teile

In der Domäne Instandhaltung wird der Store-Prozess neben dem klassischen Einsatz als Lager oder Pufferspeicher auch zur Abbildung der instand zu haltenden Anlage verwendet. Für jede Komponente einer Anlage wird ein Store-Objekt modelliert. Darin wird genau ein Teil „gelagert“. Dieses Teil ist somit das Instandhaltungsgut, das inspiziert, gewartet und bei Bedarf instand gesetzt werden muss.

Die Abnutzung des Teils an seinem Einsatzort, dem Store-Objekt, wird mittels der, im Attribut „Condition adjustment“ hinterlegten Regel definiert. Dabei ist zu beachten, dass der tatsächliche Zustand eines Teils für andere SC-Objekte nicht andauernd transparent ist, sondern nur im Rahmen von Inspektionen oder wenn eine ständige Überwachung mittels Sensoren erfolgt, bekannt wird.

5.3.3 Modellierungsklasse „Plan“

Eine Supply Chain kann zentral, durch ein Steuerungssystem, dezentral durch einzelne Anlagen oder Bearbeiter oder autonom durch bewegliche logistische Teile gesteuert werden. Häufig findet man eine Kombination aus diesen Steuerungsverfahren.

Zentrale Steuerungsaufgaben und Steuerungen von mehreren verknüpften Anlagen können mit Objekten der Klasse Plan abgebildet werden. Die Methode Wertstromdesign verwendet

ein ähnliches Konstrukt zur Abbildung von Steuerungsaufgaben, worunter Rother (Rother und Shook, 2015) den Geschäftsprozess Produktionsplanung und -steuerung (PPS) versteht.

Plan-Objekte sind Teil des Informationsflusses. Sie arbeiten ein vordefiniertes Programm ab, reagieren auf externe Ereignisse oder fragen Statusinformationen von anderen SC-Objekte ab. Sie reagieren je nach vordefiniertem Verhalten mit dem Versenden von Aufträgen. Durch die Anlage von Information flow channel-Relationen zwischen den Plan-Objekten untereinander oder zwischen Plan- und Materialfluss-Objekten werden Kommunikationskanäle dargestellt, die für den Versand von Nachrichten oder Abrufe verwendet werden können. Die Kommunikation zwischen verschiedenen Plan-Objekten untereinander kann verwendet werden, um Abstimmungs- beziehungsweise Verhandlungssituationen oder den Übergang von der zentralen zur dezentralen Steuerung abzubilden.

5.3.3.1 Attribut „Action“

In diesem Attribut wird ein aktives Steuerungsprogramm hinterlegt, das zu Beginn jeder Periode vom Simulationssystem ausgewertet wird. Das Programm wird als Regel oder Regelset definiert. Die Regeln werden in jeder Periode genau einmal durchlaufen. Aktiv bedeutet hier, im Gegensatz zu reaktiv, dass die Interaktion vom Plan-Objekt initiiert wird. Dadurch bestimmen diese Regeln maßgeblich das Steuerungsverhalten der Supply Chain. Sie können über Abrufe, den Lagerstand einzelner Store-Objekte ermitteln und per Nachrichten Aufträge versenden.

5.3.3.2 Attribut „Reaction“

Das reaktive Steuerungsprogramm das mittels Regeln in diesem Attribut definiert wird, wird immer durchlaufen, sobald eine Nachricht für das Plan-Objekt eintrifft. Sie können in einer Periode also 0..n-mal durchlaufen werden. Durch Berücksichtigung des Nachrichtentyps kann sich die Reaktion des Plan-Objekts je nach Informationstyp unterscheiden.

Durch die Trennung in aktive und reaktive Regelsets können maschinelle Verhandlungen (Softwareagenten) zwischen Supply Chain Partnern simuliert werden. Ein Partner A stößt dafür in seinem Plan-Objekt mittels aktiver Regel die Kommunikation an und sendet eine Nachricht an Partner B, dieser wertet seine reaktiven Regeln aus und antwortet. Nun wertet auch Partner A seine reaktiven Regeln aus und reagiert entsprechend. Dieser Nachrichtenaustausch erfolgt innerhalb einer Periode. Eine zukünftige Erweiterung des Konzepts könnte eine zeitverzögerte Antwort berücksichtigen.

Um Daten in späteren Perioden auswerten zu können oder Verzögerungen modellhaft darzustellen, verfügt die Regelsprache von Plan über die Möglichkeit der Speicherung und dem Zugriff auf Datenspeicher.

5.3.3.3 Attribut „Timestamp“

Wird von der Simulation mit dem jeweils aktuellen Stand der Simulationsuhr belegt. Dieser Wert wird im graphischen Modell angezeigt.

5.3.3.4 Attribut „Micro step“

Wird während der Animation mit dem jeweils aktuellen Microstep belegt. Dieser Wert wird im graphischen Modell angezeigt.

5.3.3.5 Attribut „Display“

Inhalte dieses Attributs werden während der Simulation auf der Zeichenfläche, neben dem Plan-Objekt angezeigt. Der Wert kann über einen speziellen Zuweisungsbefehl der Regeln des Plan-Objekts gesetzt werden.

5.3.3.6 Attribut „Resource“

Kann zur Referenzierung von Ressourcen, die die Steuerung durchführen, wie zum Beispiel Rollen, Services, Anwendungen verwendet werden, dient ausschließlich der Dokumentation.

5.3.3.7 Relationen von Plan-Objekten

Objekte der Klasse Plan können für sich alleine stehen oder über Objekte der Relationsklasse Information flow channel mit Objekten anderer Klassen verbunden sein. Es darf kein Material flow channel in das Objekt dieser Klasse ein- oder ausgehen. Dabei gelten folgende Kardinalitäten für die Modellierung eines Plan-Objekts:

Tabelle 9: Kardinalitäten eines Plan-Objekts

K1	es muss mindestens 1 Plan-Objekt in jedem Modell existieren
K2	es hat 0 eingehende Material flow channel
K3	es hat 0 ausgehende Material flow channel
K4	es hat 0..m eingehende Information flow channel
K5	es hat 0..m ausgehende Information flow channel
K6	ausgehende Information flow channel gehen nicht zu Objekten der Klasse Store und Transport
K7	eingehende Information flow channel kommen nicht vom Objekt der Klasse Switch

5.3.3.8 Domänenspezifische Besonderheiten

Verwendung von Plan-Objekten im Rahmen der Maintenance Supply Chain:

- In der Beschaffungs-Supply Chain von Ersatzteilen
    - Verhandlung mit Lieferanten
    - Bestellung von Ersatzteilen
    - Abruf von Leistungen von externen Dienstleistern
- In der Wartung und Inspektion
    - Festlegung der Wartungszeitpunkte
    - Koordination von Inspektionsplanung und Produktionsplanung
    - Überwachung der Lagerpolitik des Ersatzteillagers
    - Beauftragung / Benachrichtigung von externen Dienstleistern
- In der Instandsetzung
    - Zusammenstellung und Beauftragung des Instandsetzungsteams
    - Koordination aller benötigten Ressourcen
    - Entscheidung über priorisierte Bearbeitungen
    - Entscheidung über Reparatur oder Neukauf von Ersatzteilen

5.3.4 Modellierungsklasse „Transport“

Mit einem Objekt der Klasse Transport kann die Ortsveränderung von Teilen modelliert werden. Die Klasse Transport ist eine Spezialisierung der Klasse Store, sie unterscheidet sich allerdings dadurch, dass während eines Transports keine Zugriffsmöglichkeit auf Teile besteht.

Ähnlich wie Store, ist ein Transport-Objekt nicht mit einem individuellen Transportmittel gleichzusetzten, sondern mit einem Transport-Service. Der Transport eines Teils beginnt, sobald es eingelagert ist. Soll eine Menge von Teilen auf einmal transportiert werden, so muss ein Pufferlager und ein Make-Prozess, der zum gewünschten Zeitpunkt alle Teile in das Transport-Objekt bewegt, modelliert werden.

5.3.4.1 Attribut „Item type“

In einem Transport-Objekt können eine beliebige Anzahl von Teile genau eines Teiletyps transportiert werden. Die Typisierung eines Transports definiert somit eindeutig den Teiletyp der Teile, die über die ein- und ausgehenden Material flow channels des Transport-Objekts bewegt werden können. Sollen im selben Transport Teile unterschiedlichen Teiletyps bewegt werden, so müssen diese zuvor verpackt werden. Die Verpackungen müssen vom selben Typ sein, also beispielsweise ein Container für einen Seetransport.

5.3.4.2 Attribut „Transport time“

In dieses Attribut wird die Transportdauer in Perioden eingetragen. Nach Ablauf dieser Zeitdauer kann ein transportiertes Teil wieder aus dem Objekt entnommen werden. Erfolgt die Entnahme nicht sofort, verbleibt das Teil weiterhin im Transport-Prozess. Die Transportdauer stellt somit eine minimale Verweildauer eines temporären Objekts im Transport-Objekt dar.

5.3.4.3 Attribut „Condition adjustment“

Das Attribut ist identisch mit dem des Store-Objekts, siehe Attribut „Condition adjustment“, Seite 134.

5.3.4.4 Attribut „Condition history“

Das Attribut ist identisch mit dem des Store-Objekts, siehe Attribut „Condition history“ Seite 134.

5.3.4.5 Attribut „Number of items arrived“

Anzahl der Teile in diesem Transport-Objekt, die am Bestimmungsort eingetroffen sind und entladen werden können. Die Anzahl wird vom Simulationsalgorithmus ermittelt und nach jedem Entladevorgang aktualisiert. Am Ende einer Periode wird ebenfalls eine Aktualisierung durchgeführt, um alle Zugänge während dieser Periode zu berücksichtigen.

5.3.4.6 Attribut „Resource“

Kann zur Referenzierung eines physischen Transportmittels, abgebildet als Objekt im Ressourcen Modell, verwendet werden, dient ausschließlich der Dokumentation.

5.3.4.7 Attribut „Items in transport“

Dieses Attribut wird während einer Simulation vom Simulationsalgorithmus mit Werten gefüllt. Das Attribut enthält alle Teile, die sich momentan im Transport befinden und auf die nicht zugegriffen werden kann.

5.3.4.8 Attribut „Items arrived“

Dieses Attribut wird während einer Simulation vom Simulationsalgorithmus mit Werten gefüllt. Das Attribut enthält alle Teile, die am Bestimmungsort eingetroffen sind und zur Entladung bereitstehen.

5.3.4.9 Attribut „Transport cost per item“

Gesamtkosten, die für den Transport eines Teils entstehen. Bei Behältern sind das die Kosten für den gesamten Behälter. Eine Division durch enthaltene Teile erfolgt nicht.

5.3.4.10 Relationen von Transport-Objekten

Objekte der Klasse Transport können sowohl über Relationen der Klasse Material flow channel als auch über Objekte der Relationsklasse Information flow channel mit Objekten anderer Klassen verbunden sein. Dabei gelten folgende Kardinalitäten für die Modellierung eines Transport-Objekts:

Tabelle 10: Kardinalitäten eines Transport-Objekts

K1	es hat 1...n eingehende Material flow channel
K2	es hat 1..n ausgehende Material flow channel
K3	es hat keine eingehenden Information flow channel
K4	es hat 0..m ausgehende Information flow channel
K5	ausgehende Material flow channel gehen immer direkt zu Objekten der Klasse Make oder Deliver
K6	eingehende Material flow channel kommen von einem Objekt der Klasse Make, Source, Reader oder Switch

5.3.4.11 Domänenspezifische Besonderheiten

Zwischen den Objekten der Klasse Make, Source und Deliver muss für jeden eingehenden Material flow channel mindestens ein Store/Transport Objekt modelliert sein, deshalb haben diese Puffer oder Lagerprozesse in der gesamten Maintenance Supply Chain unterschiedliche Aufgaben:

- In der Beschaffungs-Supply Chain von Ersatzteilen
    - zur Berücksichtigung der Transportzeit einzelner Ersatzteile in der Wartung und Inspektion
    - Transport von Werkzeug vom Lagerort zum Einsatzort der instand zu haltenden Anlage
- In der Instandsetzung
    - Transportzeit von Ersatzteilen
    - Reisezeit von externen Spezialisten oder Know-how Trägern des Anlagen-Herstellers

Ist das instand zu haltende Gut selbst ein Verkehrsmittel, wie ein Flugzeug, das nicht immer in der Werft für Inspektionsarbeiten zur Verfügung steht, so kann der Transport-Prozess entsprechend dem Store-Objekt für immobile Anlagen als Platzhalter für mobile Anlagen verwendet werden. Die Abnutzung des Teils während seines Transports kann mittels der im Attribut „Zustandsänderung“ hinterlegten Regel definiert werden.

5.3.5 Modellierungsklasse „Source“

Source-Objekte stellen den Startpunkt des betrachteten Materialflusses dar, sie erzeugen entsprechend einer vorgegebenen Mengenverteilung Teile und speisen diese in das Materialfluss-System ein. Wie Make-Objekte generieren sie Teile, benötigen dafür aber keinen Input, beziehungsweise genauer gesagt, der vom Lieferanten für die beschafften Teile aufgewendete Input ist nicht Inhalt der Betrachtung des Modells. Sie geben die dynamischen Teile direkt und unverzüglich an die adjazenten Store-Objekte weiter. Die Teile können in weiterer Folge von Produktionsprozessen (Make-Objekten) des SCN Modells verarbeitet werden.

5.3.5.1 Attribut „Sourcing program“

Ein Source-Prozessschritt liefert pro Auftrag eine Kombination verschiedener Teiletypen in jeweils definierter Menge. Die Lieferung erfolgt verzögerungsfrei und jede Bestellung wird vollständig erfüllt. Wie beim Make-Prozess erfolgt die Definition der zu liefernden Teiletypen und Mengen durch ein Programm, dem sogenannten Einkaufsprogramm. Die Syntax des Einkaufsprogramms wird im Kapitel Implementierung beschrieben.

5.3.5.2 Attribut „Order“

Dieses Attribut enthält die Anzahl der in einer Periode auszuführender Beschaffungsaufträge. Eine Auftragshöhe von “0“ bedeutet, dass das Source-Objekt in dieser Periode keine dynamischen Teile bereitstellt.

Der Wert des Attributs ist typischerweise während einer Simulation nicht stabil, sondern kann von anderen Objekten mittels Nachrichten oder vom Objekt selbst, entsprechend einer Regel, verändert werden:

Ständige Lieferung. Durch Angabe einer Konstanten im Attribut

Lieferung zu definierten Zeitpunkten. Durch Definition einer zeitpunktbezogenen Regel

Lieferung in Abhängigkeit des Zustandes anderer SC-Objekte. In dem beispielsweise mittels eines Information flow channels eine Abfrage über den Lagerstand eines Store-Objekts gemacht wird.

Ist der resultierende Wert im Attribut „Order“ größer 1, so entspricht diese Zahl einem Multiplikator, der auf das Einkaufsprogramm angewendet wird. Es wird somit ein, dem Multiplikator entsprechendes Vielfaches der Teile geliefert.

5.3.5.3 Attribut „Order history“

Dieses Attribut wird während einer Simulation vom Simulationsalgorithmus mit Werten gefüllt. Für jede Periode wird die Anzahl der Beschaffungsaufträge in diesem Attribut erfasst. Andere Objekte können per Abruf auf dieses Attribut zugreifen und diese Information für Steuerungsaufgaben verwenden.

5.3.5.4 Attribut „Cost function“

Es kann eine Regel definiert werden, die die Kosten der Bestellung ermittelt. Damit können beispielsweise Rabatte bei Bestellung größerer Mengen im Modell oder Preissteigerungen über die Zeit abgebildet und zur Steuerung herangezogen werden.

5.3.5.5 Relationen von Source-Objekten

Objekte der Klasse Make können sowohl über Relationen der Klasse Material flow channel als auch über Objekte der Relationsklasse Information flow channel mit Objekten anderer Klassen verbunden sein. Da Source-Objekte am Beginn des Materialflusses stehen, können sie keine eingehenden Material flow channel haben. Dabei gelten folgende Kardinalitäten für die Modellierung eines Source-Objekts:

Tabelle 11: Kardinalitäten eines Source-Objekts

K1	es hat 0 eingehende Material flow channel
K2	es hat 1 ausgehenden Material flow channel
K3	es hat 0..m eingehende Information flow channel
K4	es hat 0..m ausgehende Information flow channel
K5	der ausgehende Material flow channel führt zu einem Objekt der Klasse Store, Transport, Reader oder Switch

Da nur maximal ein ausgehender Material flow channel erlaubt ist, muss für den Fall, dass aus dem Source-Objekt Teile unterschiedlichen Teiletyps resultieren, zwischen dem Source-Objekt und den nachfolgenden Store-Objekten ein Switch gesetzt werden.

5.3.5.6 Domänenspezifische Besonderheiten

In der Maintenance-Supply Chain werden Objekte der Klasse Source zum Scoping des Beobachtungssystems in Richtung externe und interne Lieferanten verwendeten:

- in der Beschaffungs-Supply Chain von Ersatzteilen
    - Beschaffung von Ersatzteilen
    - Beschaffung von Werkzeugen

Ein anderes Einsatzgebiet von Source-Objekten im SCN-Modell ist die initiale Befüllung von Store-Objekten, um kurzzeitig ein Eingeschwungenes System zu erreichen. Dafür wird an die betreffenden Store-Objekte jeweils ein Source-Objekt gehängt, das in der Auftragshöhe eine zeitgesteuerte Regel enthält und nur in den ersten Perioden die erforderlichen Teile in die Lager liefert.

5.3.6 Modellierungsklasse „Deliver“

Deliver-Objekte stehen am Ende des Materialflusses, sie verbrauchen Teile und entziehen diese dem Materialfluss-System. Bei Supply Chain Network Modellen, die bis an die Unternehmensgrenzen modelliert werden, bilden Objekte der Klasse Deliver den Lieferprozess an einen externen Kunden ab.

Wie Make-Objekte konsumieren sie Teile, liefern aber keinen Output, genauer gesagt, die Verfolgung der Teile, die der Kunde erhält, sind nicht Inhalt der Betrachtung des Modells.

5.3.6.1 Attribut „Delivery program“

Ein Deliver-Prozessschritt benötigt pro Auftrag eine Kombination verschiedener Teiletypen in jeweils definierter Menge. Wie beim Source- und Make-Prozess erfolgt die Definition der an den Kunden zu liefernden Teiletypen und Mengen durch ein Programm, dem sogenannten Lieferprogramm. Die Syntax des Lieferprogramms wird im Kapitel 7 Implementierung beschrieben.

5.3.6.2 Attribut „Order“

Dieses Attribut enthält die Anzahl der in einer Periode zu liefernden Aufträge. Eine Auftragshöhe von “0“ bedeutet, dass das Deliver -Objekt in dieser Periode keine dynamischen Teile abnimmt. Ist die Auftragshöhe „1“, so werden die im Lieferprogramm angegebenen Teile aus den jeweiligen adjazenten Store-Objekten der spezifizierten Menge entnommen. Der Wert des Attributs ist typischerweise während einer Simulation nicht stabil, sondern kann von

anderen Objekten mittels Nachrichten oder vom Objekt selbst, entsprechend einer Regel, verändert werden:

Ständige Abnahme. Durch Angabe einer Konstanten im Attribut

Abnahme zu definierten Zeitpunkten. Durch Definition einer zeitpunktbezogenen Regel

Abnahme in Abhängigkeit des Zustandes anderer SC-Objekte. In dem beispielsweise mittels eines Information flow channels eine Abfrage über den Lagerstand eines Store-Objekts gemacht wird.

Ist der resultierende Wert im Attribut „Order“ größer 1, so entspricht diese Zahl einem Multiplikator, der auf das Lieferprogramm angewendet wird. Es wird somit ein, dem Multiplikator entsprechendes Vielfaches der Teile geliefert.

Ein Auftrag, der nicht erfüllt werden kann, weil in den vorgelagerten Store-Objekten nicht genügend Teile lagern, verfällt. Teilaufträge werden nicht ausgeführt.

5.3.6.3 Attribut „Delivery history – to be“

Dieses Attribut wird während einer Simulation vom Simulationsalgorithmus mit Werten gefüllt. Für jede Periode wird die Anzahl der durchzuführenden Lieferaufträge in diesem Attribut erfasst. Andere Objekte können per Abruf auf dieses Attribut zugreifen und diese Information für Steuerungsaufgaben verwenden.

5.3.6.4 Attribut „Delivery history – as is“

Dieses Attribut wird während einer Simulation vom Simulationsalgorithmus mit Werten gefüllt. Für jede Periode wird die Anzahl der tatsächlich durchgeführten Lieferaufträge in diesem Attribut erfasst. Andere Objekte können per Abruf auf dieses Attribut zugreifen und diese Information für Steuerungsaufgaben verwenden.

5.3.6.5 Attribut „Revenue function“

Es kann eine Regel definiert werden, welche die Erlöse aus Sicht der Supply Chain für einen gelieferten Auftrag ermittelt. Damit können beispielsweise Rabatte bei Lieferung größerer Mengen im Modell oder Preissteigerungen über die Zeit abgebildet und zur Steuerung herangezogen werden.

5.3.6.6 Relationen von Deliver-Objekten

Objekte der Klasse Deliver können sowohl über Relationen der Klasse Material flow channel als auch über Objekte der Relationsklasse Information flow channel mit Objekten anderer

Klassen verbunden sein. Da Deliver-Objekte am Ende des Materialflusses stehen, können sie keine ausgehenden Material flow channel haben. Dabei gelten folgende Kardinalitäten für die Modellierung eines Deliver-Objekts:

Tabelle 12: Kardinalitäten eines Deliver-Objekts

K1	es hat 1..n eingehende Material flow channel
K2	es hat 0 ausgehende Material flow channel
K3	es hat 0..m eingehende Information flow channel
K4	es hat 0..m ausgehende Information flow channel
K5	der eingehende Material flow channel kommt aus einem Objekt der Klasse Store oder Transport

5.3.6.7 Domänenspezifische Besonderheiten

In der Maintenance-Supply Chain werden Objekte der Klasse Deliver zum Scoping des Beobachtungssystems in Richtung externe und interne Kunden verwendeten:

- Im Falle einer Instandsetzung
    - Rücknahme ausgebauter Teile
    - Rückreise von externen Spezialisten, deren Betrachtung nicht mehr im Scope ist

Ein weiterer wichtiger Anwendungsfall von Deliver-Objekten ist die Abbildung von Schwund. Im Unterschied zu der in den Store-Objekten abbildbaren Zustandsveränderung, die nur Auswirkung auf die Qualität eines Teils hat, entfernt das Deliver-Objekt ein Teil aus dem Store-Objekt vollständig. Die Steuerung, wann, beziehungsweise wie häufig das passiert, kann über eine zufallsgesteuerte Regel erfolgen.

5.3.7 Modellierungsklasse „Reader“

Objekte der Klasse Reader sind Übergangspunkte von Informations- und Materialfluss. Mittels Readern können Daten aus den beweglichen logistischen Teilen ausgelesen werden oder Daten auf einzelne dynamische Teile aufgebracht werden. Demgemäß kann es sich sowohl um eine Lese- als auch Schreibvorrichtung handeln. Mittels Readern können alle Auto-ID-Verfahren abgebildet werden. Ein Reader gibt keine technologische Implementierung vor, es kann sich unter anderem um einen RFID-Leser oder einen Drucker, der Barcode oder OCR-lesefähige Texte aufbringt, handeln.

Die Information wird also nicht durch einen Bearbeitungsvorgang gewonnen, sondern sie wird direkt aus oder von einem individuellen Teil gelesen. Dynamische Teile durchlaufen einen Reader verzögerungsfrei. Bildet ein Reader ein RFID-System ab, so ist er in der Lage im Bulk einzelne verpackte Teile zu identifizieren und die ausgelesene Information in den Informationsfluss einzuspeisen. Nachdem die Teile einen Reader durchlaufen haben werden sie sofort im Materialfluss zum nachfolgenden SC-Objekt bewegt.

5.3.7.1 Attribut „Stack“

Dieses Attribut wird während einer Simulation vom Simulationsalgorithmus mit Werten gefüllt. Jedes Teil, das am Reader vorbeigeführt wird, wird hier eingetragen. Während der Simulation wird dieses Attribut am oberen Rand des Objekts visualisiert.

5.3.7.2 Attribut „Action“

Im Action-Attribut wir ein Regelset hinterlegt, das für jedes Teil, das am Reader vorbeigeführt wird individuell ausgewertet wird. Dabei können Informationen auf das Teil aufgebracht werden, wie zum Beispiel ein Haltbarkeitsdatums, oder Daten aus dem Teil ausgelesen werden. Je nach Einsatzzweck kann ein Auslesevorgang einen Versand von Nachrichten über einen Information flow channel auslösen.

5.3.7.3 Attribut „Reaction“

Im Reaction-Attribut wir ein Regelset hinterlegt, das für jedes Teil, das am Reader vorbeigeführt wird individuell ausgewertet wird. Die Ausführung des Regelsets erfolgt zeitlich immer nach der Auswertung der Aktions-Regeln. Es kann somit auf deren Ergebnisse oder auf eine über einen Information flow channel gesendete Nachricht, reagieren.

5.3.7.4 Attribut „Resource“

Kann zur Referenzierung eines physischen Readers, abgebildet als Objekt im Resource Model verwendet werden, dient ausschließlich der Dokumentation.

5.3.7.5 Relation von Reader-Objekten

Objekte der Klasse Reader können sowohl über Relationen der Klasse Material flow channel als auch über Objekte der Relationsklasse Information flow channel mit Objekten anderer Klassen verbunden sein. Dabei gelten folgende Kardinalitäten für die Modellierung eines Reader-Objekts:

Tabelle 13: Kardinalitäten eines Reader-Objekts

K1	es hat 1..n eingehende Material flow channel
K2	es hat 1 ausgehenden Materialfluss flow channel
K3	es hat 0..m eingehende Information flow channel
K4	es hat 0..m ausgehende Information flow channel
K5	der eingehende Material flow channel kommt aus einem Objekt der Klasse Source, Make, Reader oder Switch
K6	der ausgehende Material flow channel führt zu einem Reader, Switch, Store oder Transport-Objekt

5.3.7.6 Domänenspezifische Besonderheiten

In der Maintenance Supply Chain wird der Zustand eines Teils mittels der, im Store-Objekt hinterlegbaren Regel zur Zustandsänderung abgebildet und der repräsentative quantitative Wert in einer Variablen des Teils gespeichert. Zur Abbildung der im Wartungsprozess notwendigen Inspektionstätigkeit wird eine Sequenz von einem Make-Objekt und einem Reader-Objekt modelliert. Das Make-Objekt bildet hierbei die zeitliche Dauer des Teileaus- und Teileeinbaus der Inspektion und deren Ressourcenbedarf ab, während der dazugehörige Reader den Zustandswert des Teils ausliest und diese Daten an eine Steuerung übergibt. In weiterer Folge kann das inspizierte Teil wieder eingebaut oder ausgesteuert werden.

5.3.8 Modellierungsklasse „Switch“

Mit Objekten der Klasse Switch können Verzweigungen in Material flow channels abgebildet werden. Sollen Teile je nach Typ oder Bestimmung unterschiedliche Wege im Materialfluss durchlaufen, so ist eine Verzeigung notwendig.

Alle aus einem Objekt ausgehenden Material flow channel müssen eine disjunkte Nummer im Attribut „Index“ aufweisen. Das Switch-Objekt entscheidet nicht autonom über die Steuerung der einzelnen Teile. Die Teile werden entweder per eintreffender Nachrichten gelenkt oder aufgrund ihres Teiletyps automatisch einem indizierten Material flow channel zugewiesen.

5.3.8.1.1 Modus: typbezogen

Die Indexe der ausgehenden Material flow channels entsprechen den nummerischen Identifizierern der Teiletypen, die über den Switch bewegt werden können. Ist ein Switch

hinter einem Make-Objekt platziert das gemäß des Produktionsprogramms Teile mit der Typ-ID 3 und 4 produziert, so müssen aus dem Switch-Objekt genau zwei Material flow channels einer mit Index 3 und der andere mit Index 4 austreten.

5.3.8.1.2 Modus: nachrichtbezogen

Die Steuerungsinformation, welchen Weg ein Teil nehmen soll, wird über eintreffende Nachrichten übermittelt. Für jedes dynamische Teil kann über einen Information flow channel eine Nachricht eintreffen, die spezifiziert, über welchen alternativen Material flow channel es bewegt werden soll. Trifft ein Teil in einem Switch ein, ohne dass eine individuelle Nachricht vorliegt, so wird es über den Defaultpfad geleitet. Dieser Defaultpfad kann ebenfalls über eine Nachricht verändert werden.

5.3.8.2 Attribut „Resource“

Kann zur Referenzierung eines physischen Materialflussobjekts, abgebildet als Objekt im Resource Model, verwendet werden, dient ausschließlich der Dokumentation.

5.3.8.3 Relationen von Switch-Objekten

Objekte der Klasse Switch können sowohl über Relationen der Klasse Material flow channel als auch über Objekte der Relationsklasse Information flow channel mit Objekten anderer Klassen verbunden sein. Dabei gelten folgende Kardinalitäten für die Modellierung eines Switch-Objekts:

Tabelle 14: Kardinalitäten eines Switch-Objekts

K1	es hat 1..n eingehende Material flow channel
K2	es hat 1..n ausgehende Material flow channel
K3	es hat 0..m eingehende Information flow channel
K4	es hat 0 ausgehende Information flow channel
K5	der eingehende Material flow channel kommt aus einem Objekt der Klasse Source, Make, Reader oder Switch
K6	die ausgehenden Material flow channels führen zu Reader, Switch, Store oder Transport-Objekten

5.3.8.4 Domänenspezifische Besonderheiten

Das Switch-Objekt kann beispielsweise zur Aussteuerung von ausgewechselten Ersatzteilen oder Trennung von Verpackungen und Teilen verwendet werden.

5.3.9 Beziehungstyp „Material flow channel“

Relationen vom Typ Material flow channel setzen jeweils zwei statische Supply Chain Objekte zueinander in Beziehung. Die gerichtete Beziehung drückt aus, dass vom ausgehenden Objekt ein dynamisches Teil zum eingehenden Objekt bewegt werden kann.

Technisch kann dies durch ein Förderband, einen Gabelstapler oder einfach durch eine Handbewegung eines Maschinenbedieners oder Serviceingenieurs erfolgen. Die Transportzeit eines auf einem Material flow channel bewegten Teils ist 0. Beansprucht der Transport eine relevante Durchführungszeit, so muss in den Materialfluss ein Objekt vom Typ Transport eingefügt werden.

Die Abbildung eines Material flow channels zwischen zwei Objekten ist noch kein hinreichender Hinweis auf die tatsächliche Bewegung von Material über diese Verbindung während der betrachteten Periode. Dies hängt maßgeblich vom Verhalten des Systems im Zeitverlauf ab.

5.3.9.1 Attribut „Index“

Wird ausschließlich in Material flow channel-Konnektoren verwendet, die aus Objekten der Klasse Switch münden. Die Indizes der aus einem Switch ausgehenden Relationen müssen eindeutig sein.

5.3.10 Beziehungstyp „Information flow channel“

Relationen vom Typ Information flow channel setzen jeweils zwei statische Supply Chain Objekte zueinander in Beziehung. Die gerichtete Beziehung drückt aus, dass vom ausgehenden Objekt ein dynamisches Informationsobjekt, wie eine Nachricht oder ein Abruf, zum eingehenden Objekt gesendet werden kann. Die sofortige Antwort auf einen Abruf wird vom adressierten Objekt über denselben Kanal zurückgegeben, auch wenn dies entgegen der Richtung der Relation ist.

Dies kann durch ein technisches Medium, eine Softwarelösung (wie zum Beispiel Web-Services, EDI, Email) Papier oder Stimme erfolgen. Die Zeit, in der eine Nachricht über einen Information flow channel übermittelt wird ist 0. Soll eine längere Übermittlungsdauer

abgebildet werden, muss dies mittels eines Plan-Objekts und dessen Datenspeicherfunktion modelliert werden.

Die Abbildung eines Information flow channels zwischen zwei Objekten ist noch kein hinreichender Hinweis auf die tatsächliche Übermittlung von Informationen über diese Verbindung während der betrachteten Periode. Dies hängt maßgeblich vom Verhalten des Systems im Zeitverlauf ab und der von SC-Objekten tatsächlich versendeten Nachrichten beziehungsweise getätigten Abrufe.

5.3.10.1 Attribut „Index“

Wird verwendet um den Information flow channel in einer Regel eindeutig zu identifizieren und zu referenzieren. Die Indizes der aus einem Objekt ausgehenden Relationen müssen eindeutig sein.

5.4 Modelltyp „Resource Model“

In der Literatur werden einige Metamodelle beziehungsweise Ontologien beschrieben, die Unternehmensressourcen im Produktions- oder Supply Chain Umfeld betrachten. Einen guten Überblick bieten Grubic et al. (Grubic und Fan, 2010). Die von ihnen vorgestellten Ontologien beschreiben hauptsächlich die strategische Ebene. Demensprechend ist die Granularität der Modelle relativ grob, dennoch reicht sie aus, um Kandidaten für Klassen des Modelltyps Ressource Model zu extrahieren. Im Folgenden werden einige Beispiele für ressourcenbezogenen Ontologien aufgeführt und bezüglich deren Anwendbarkeit für die Methode SIMchronization besprochen. Anschließend wird das aus diesem Input resultierende Metamodell für das Ressource Model vorgestellt.

5.4.1 Ontologie mobiler Betriebsmittel

Jendoubi (Jendoubi, 2007) stellt in seiner Arbeit eine Ontologie mobiler Betriebsmittel vor. Er definiert Betriebsmittel als „Anlagen, Geräte und Einrichtungen, die der betrieblichen Leistungserstellung dienen“. Mobile Betriebsmittel sind demgemäß Werkzeuge, Vorrichtungen oder Messmittel die besonders auch in der Domäne Instandhaltung häufig anzutreffen sind. Das Klassifikationsschema nimmt seinen Ausgang bei mobilen Objekten, die zu „mobile Fabrik Ressourcen“ auf die betrachtete Domäne spezialisiert werden. Die daraus klassifizierten, zentralen Betrachtungsgegenstände sind Werkzeuge („Tools“), Vorrichtungen beziehungsweise Halterungen („Fixture“) und Messmittel („Measurement“). Werkzeuge und Vorrichtungen können aus mehreren Teilen zusammengesetzt sein. Zur Umsetzung des Materialflusses sieht die Ontologie Transport Ressourcen („Resource Transport“) und

Lagerplätze („Storage“) vor, die jeweils Behälter („Tool Box“) zum Transport aufnehmen können.

Die in der Ontologie beschriebene Domäne der mobilen Betriebsmittelversorgung ist der, in dieser Arbeit beschriebenen Domäne der Instandhaltung sehr ähnlich, deshalb sind alle enthaltenen Klassen potentielle Kandidaten für das Metamodell des Ressourcenmodels.

Die Taxonomie enthält jedoch keine bearbeiterbezogenen Klassen wie zum Beispiel Werker oder Rolle. Außerdem fällt die Zuordnung von IT-Systemen wie Services, Anwendungen oder Hardware schwer.

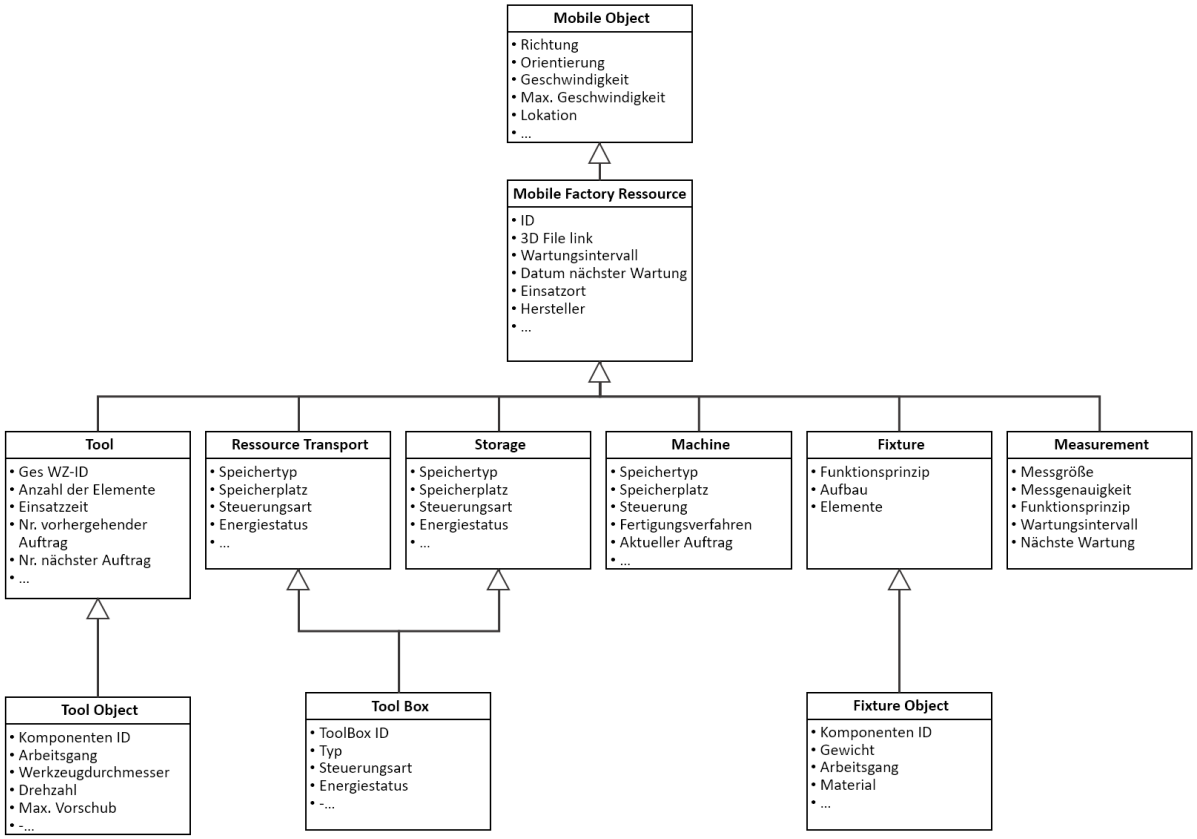


Abbildung 49: Auszug aus Ontologie mobiler Betriebsmittel (Jendoubi, 2007)

5.4.2 Onto-SCM

Onto-SCM wurde von Ye et al. (Ye et al., 2008a) als Plansprache entwickelt, um den Austausch zwischen heterogenen Informationssystemen in einer Supply Chain zu ermöglichen beziehungsweise zu erleichtern (Grubic und Fan, 2010).

Die Ontologie ist modular aufgebaut, ein Teil befasst sich mit Supply Chain Ressourcen, deren Attribute und Beziehungen in der SCM Domäne beschrieben werden. Ein anderer Teil befasst sich mit den Teilen, auf die in dieser Arbeit im Kapitel 5.5.1 eingegangen wird. Abbildung 50

zeigt den ersten Detaillevel der eine Ressource in die Subklassen ‚Production_Resource', ‚Storage_Resource', ‚Transportation_Resource' and ‚Human_Resource' unterteilt. Ye et al. führen aus, dass diese Klassen noch tiefer spezialisiert werden können, um spezielle Maschinentypen und Rollen von Mitarbeitern unterscheiden zu können. Als Beispiel führen sie die Ressourcen Fahrzeug, Container und Förderband als Spezialisierung der Klasse ‚Transportation_Resource' an. Die Besonderheit der Klasse ‚Human_Resource' ist, dass sie verschiedenen Rollen zugeordnet werden kann. Außerdem wird hervorgehoben, dass die ‚Human_Resource' in der Lage ist, Ressourcen anderer Klassen zu benutzen, zu steuern oder zu warten.

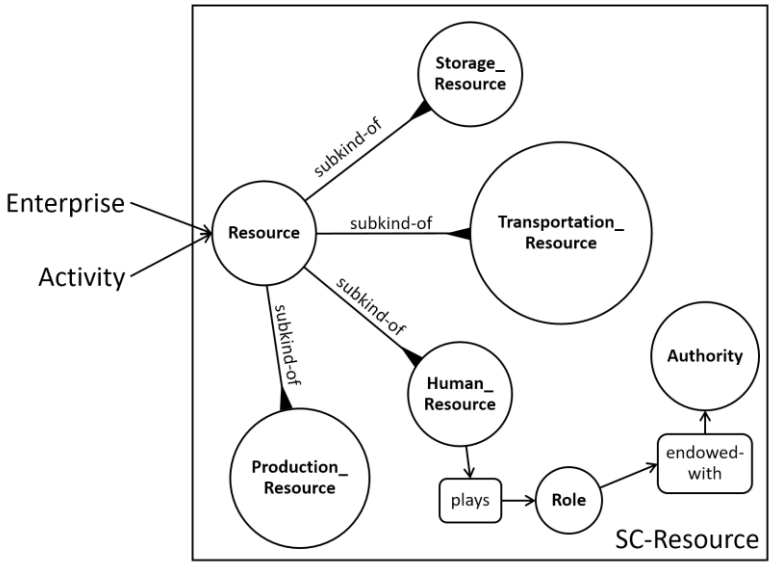


Abbildung 50: Resource Model der Onto-SCM (Ye et al., 2008b)

Für ein Ressourcenmodell, das speziell in der Domäne Instandhaltung Anwendung finden soll, sollte die Production_Resource etwas genereller formuliert werden, um beispielsweise auch Messgeräte oder mobile Handhelds verständlich als Ressourcen abbilden zu können. Ebenfalls als Production_Resource müssen in auf Basis von Onto-SCM IT-Systeme abgebildet werden, was der gewünschten Transparenz der Methode und intuitiven Handhabung entgegenspricht.

5.4.3 SCOntology

Vegetti (Vegetti et al.) streichen in ihrer Arbeit die Wichtigkeit der Informationslogistik in Kombination mit einem effizienten Supply Chain Management heraus. Zur Unterstützung des Datenaustauschs zwischen Steuerungs- und Produktions-Systemen und zur Verfeinerung des Metamodells schlagen sie die Verwendung des ANSI / ISA S95.00.01-2000 Enterprise -

Control System Integration Standards, auch bekannt als ISA 95 (Greeff und Ghoshal, 2004), vor. Dementsprechend werden Ressourcen in der SCOntology in ‚Information Resources' und ‚Material Resources' unterteilt, siehe Abbildung 51. Unter Material Ressourcen fallen Material und Energie, Personal, Equipment und Prozess Segmente. Prozess Segmente sind dabei logische Gruppierungen von Personalressourcen, Geräte, Werkzeuge und Instrumente und Material, um einen Prozessschritt durchzuführen.

Material wird in der Domäne Instandhaltung als bewegliches logistisches Gut aufgefasst, dessen Weg durch den Materialfluss in Beziehung zum Informationsfluss analysiert wird. Um Engpässe und Lagerstände analysieren zu können, ist es allerdings nicht ausreichend, das benötigte Material als beschreibendes Attribut an Prozessschritte zu annotieren. Teile werden in der Methode als eigenständige dynamische Objekte betrachtet und werden im Item Model näher beschrieben. Der Energieverbrauch steht nicht im Mittelpunkt der Betrachtung der Methode SIMchronization[66]. Ressourcen werden bei der Bearbeitung von Prozessen verbraucht oder zumindest für eine bestimmte Zeit belegt. Dies trifft auf Information nicht zu, deshalb wird in der Methode SIMchronization die Information nicht als Ressource betrachtet und deren Beschreibung in Modellen des Modelltyps *Information Model* vorgenommen.

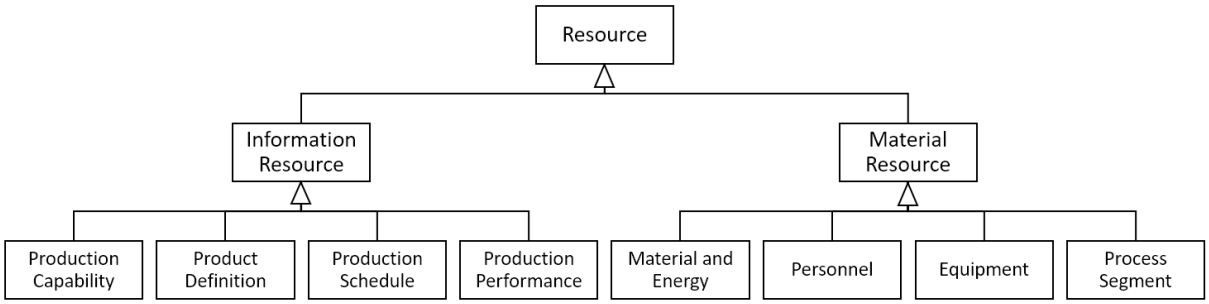


Abbildung 51: S95 Object Model (Vegetti et al., 2005)

5.4.4 TOVE Ontology

TOVE (Fox et al., 1996) wurde im Rahmen des Toronto Virtual Enterprise Projects entwickelt und ist eine bekannte Unternehmensontologie, die am Beispiel eines Produzenten für Tischlampen aufgebaut wurde. Die Ontologie ist aber generisch und für viele Domänen verwendbar. TOVE besteht aus mehreren Teil-Ontologien, die verschiedene Bereiche eines Unternehmens abdecken. Nicht alle in TOVE beschriebenen Aspekte sind für die Betrachtung des Supply Chain Managements direkt relevant, von besonderem Interesse hier ist jedoch die Ausgestaltung der Ressourcen Ontologie.

[66] Ist aus Darstellungsgründen ein Verweis auf eine bestimmte Energieform oder Quelle notwendig, so kann dies mittels einer Referenz auf die Ressource geschehen.

Im TOVE Kontext ist eine Ressource eine Rolle, die ein Objekt beziehungsweise eine Entität in Bezug auf eine Aktivität einnimmt. Eine Ressource muss also eine Assoziation zu einer Aktivität haben und zum Zeitpunkt der Ausführung der Aktivität unbelegt sein (TOVE Ontology Project, 2011a).

Rollen, die Objekte einnehmen können, um zu Ressourcen einer Aktivität zu werden, sind das „Raw material“ (Ausgangsmaterial), „Product“ (produziertes Gut), „Facility“ (Maschinen), „Tool“(Werkzeuge), „Operator“ (Person die Aktivität ausführt) und „Space“ (Arbeitsplatz/ -fläche).

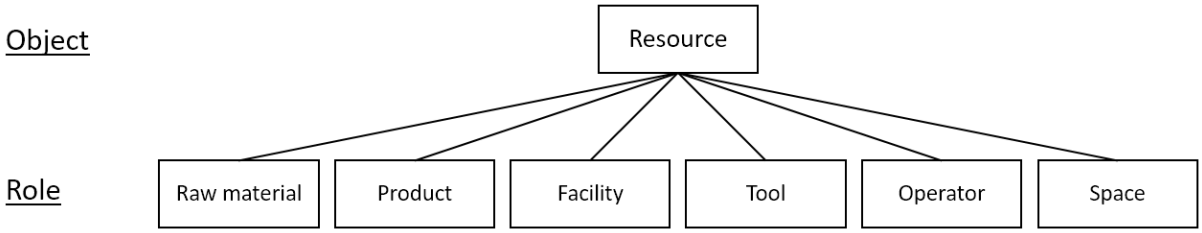


Abbildung 52: Taxonomie der Rollen (TOVE Ontology Project, 2011b)

Auch wenn in Abbildung 52 nicht enthalten, kann laut der Definition auch Elektrizität und das Kapital (wenn bei der Aktivitätsdurchführung verbraucht) oder die Information als eine Ressource aufgefasst werden (Fadel et al., 1994, S. 2). Im vorgestellten Stand von SIMchronization werden Geld- oder Kapitalflüsse nicht betrachtet. Es steht dem Modellierer aber frei, Kapital und Elektrizität als Ressource zu modellieren und den entsprechenden Aktivitäten zuzuordnen. Eine quantitative Auswertung ist allerdings nicht vorgesehen. Information wird nicht als Ressource betrachtet.

5.4.5 IDEON Ontology

Madni et al. (Madni et al., 2001) stellen mit der IEON Ontology eine Ontologie vor, die eine Grundlage für die Steuerung der Zusammenarbeit von virtuellen Unternehmen und Netzwerken von Unternehmen bietet. Sie besteht aus den vier Teilbereichen:

- Enterprise Context View
- Enterprise Organizational View
- Process View
- Resource/Product View

Unter einer Ressource wird in der IDEON Ontologie ein Objekt verstanden, das zur Durchführung einer Aktivität beziehungsweise eines Prozesses benötigt wird, eine Ressource kann aber auch das Ergebnis eines Prozesses sein.

In der Resource/Product View werden Ressourcen zunächst zu Material- und menschlichen Ressourcen spezialisiert. Eine Material Ressource kann aus weiteren Material Ressourcen

bestehen. Material Ressourcen werden in Informationsprodukte und physisches Material unterschieden. Bei einem Informationsprodukt kann es sich um einen Plan, ein Informationssystem, ein Dokument oder eine Dokumentenvorlage handeln. Die Human Resource ist entweder eine Rolle oder eine Person, die eine Rolle einnehmen kann.

Alle diese Ressourcen werden, sofern sie aus Prozessen des Unternehmens entstammen, in IDEON als Produkte des Unternehmens gesehen. Auch die Mitarbeiter, die durch Schulungsmaßnahmen eine „Wertsteigerung“ erlangen können, sind Produkte des Trainingsprozesses.

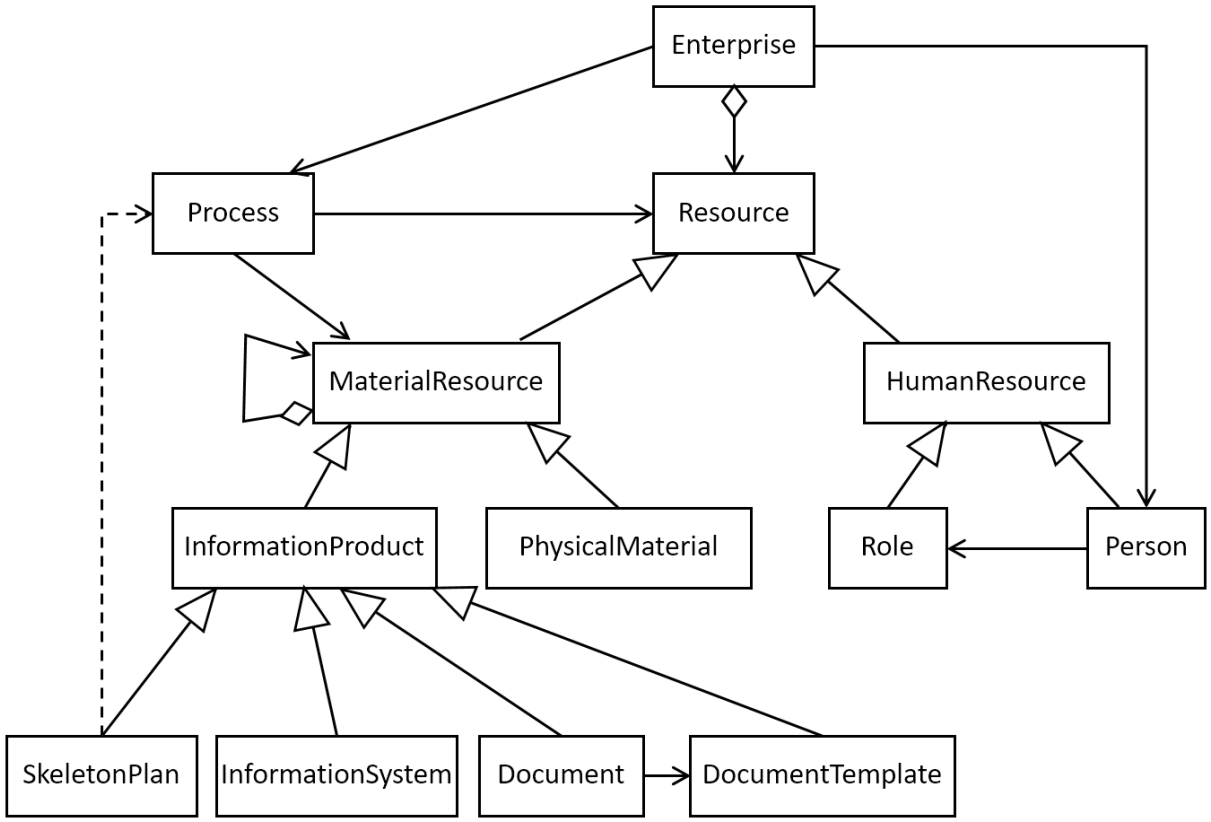


Abbildung 53: IDEON Ontology (Madni et al., 2001)

In der Methode SIMchronization sind Ergebnisse eines Prozessschritts niemals Ressourcen, sondern immer dynamische Objekte, also Material das im Materialfluss weiter bewegt wird oder Informationsobjekte als Ergebnis eines Steuerungsprozesses.

Es sind zwar alle Elemente, die im Resource Modell abgebildet sind, in der Ontologie enthalten, allerdings scheint die Einordnung etwas willkürlich. Da die Ontologie auf virtuelle Unternehmen abzielt, mag es adäquat sein, dass klassische Ressourcen wie Anlagen, Maschinen, Transportmittel nicht aufgenommen wurde.

5.4.6 Metamodell des Ressourcenmodells

Nach Studium einiger Ontologien aus dem Supply Chain Management Umfeld lässt sich folgende Schlussfolgerungen für den Aufbau des Metamodells des Ressourcenmodells der Methode SIMchronization ziehen:

- Maßgeblich für die Gestaltung des Metamodells ist die Zielsetzung der, mit der Modellierung verbundenen Methode.
- Der in der Literatur angewandte Detaillierungsgrad für Supply Chain Ontologien ist als relativ grob anzusehen. Dies resultiert aus der primären strategischen Ausrichtung der vorgestellten Ontologien. Für die Anwendung von SIMchronization ist es ausreichend, im Metamodell eine grobe Klassifikation von Ressourcentypen vorzunehmen.
- Wenngleich SIMchronization für die Domäne Instandhaltung konzipiert wurde, erscheint es auf oberster Ebene des Metamodells nicht erforderlich, domänenspezifische Klassen einzuführen. Eine Erweiterung kann durch den Anwender domänenspezifisch oder in Bedacht auf die zu erreichenden Projektziele erfolgen.
- In dem in dieser Arbeit vorgestellten Anwendungsmodell der Methode wird das Ressourcenmodell ausschließlich zur einheitlichen Dokumentation der Prozess-Schritte des Supply Chain Network Modells verwendet. Auf eine klassenspezifische Attributierung wird aufgrund der Heterogenität der möglicherweise auftretenden Ressourcentypen verzichtet.

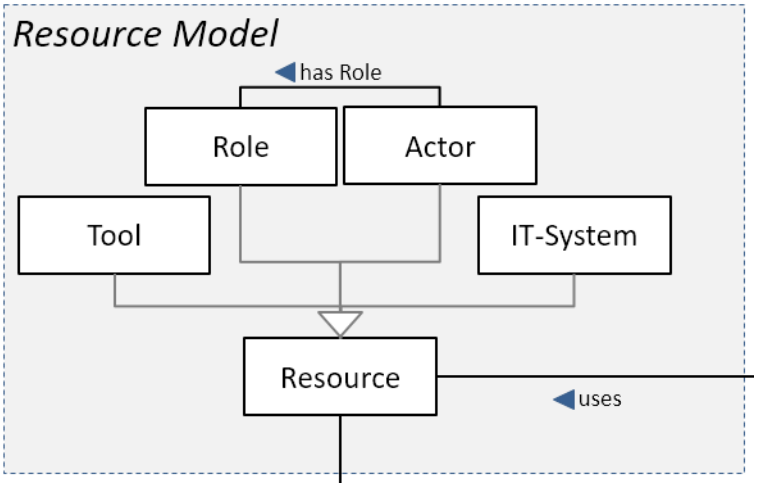


Abbildung 54: Metamodell des Resource Models der Methode SIMchronization

Die Taxonomie einer Ressource erfolgt auf folgende Weise.

5.4.7 Modellierungsklasse „Resource“

Superklasse für alle Ressourcen-Typen im Modelltyp „Resource Model“. Kann für alle Ressourcen verwendet werden, die nicht unter eine der folgenden Spezialisierungen fallen.

5.4.8 Modellierungsklasse „Tool“

Unter Tools oder Arbeitsmittel werden alle Anlagen, Maschinen, Werkzeuge, Transportmittel und sonstige Geräte zusammengefasst, die direkt oder indirekt daran beteiligt sind, den Prozess auszuführen. IT-Systeme stellen zwar auch ein Arbeitsmittel dar, sie werden aber aufgrund der vielen, sehr spezifischen Anwendungsszenarien, die einer speziellen Modellierung bedingen, als eigener Objekttyp abgebildet der hier nicht detaillierter spezifiziert wird.

5.4.9 Modellierungsklasse „Actor“

Unter Akteure werden Personen verstanden, die Prozessschritte bearbeiten oder verantworten. Je nach Größe und Mitarbeiteranzahl des untersuchten Bereichs kann ein aus der Klasse Akteur instanziiertes Objekt eine Person oder eine Position beziehungsweise Stelle sein. Der Erfahrung des Autors nach, wird aus mitbestimmungsrechtlichen Gründen gerade im deutschsprachigen Raum bei einer software-gestützten Modellierung oft darauf verzichtet, Klarnamen von Personen in das Datenbanksystem einzupflegen. Ein Akteur kann 0..n Rollen zugeordnet sein.

Auch wenn konzeptuell ein IT-System ein Akteur in einem Supply Chain Network sein kann, wird aus Gründen der Transparenz und Verständlichkeit des Modells darauf verzichtet und für IT-Systeme eine eigenen Klassifikation im Metamodell vorgesehen.

5.4.10 Modellierungsklasse „Role“

Eine Rolle definiert eine bestimmte Menge von Charakteristika, wie beispielsweise Qualifikationen oder Kompetenzen (Junginger, 2001, S. 25). Eine Rolle kann 0..n Akteuren zugeordnet werden.

5.4.11 Modellierungsklasse „IT-System“

IT-System können beispielsweise Web-Services, Hardware, Anwendungen, Schnittstellen sein, aber nicht Informationsobjekte, wie Nachrichten, Abrufe, Dokumente, diese werden im Information Model abgebildet.

5.4.12 Attribute der Klassen des „Resource Model“

Aufgrund der individuellen Erweiterbarkeit des Metamodells des Resource Model werden von der Methode nur die Attribute „Name“ und „Description“ vorgesehen.

5.4.13 Beziehungstyp „Has role“

Objekte der Klasse Actor können über Relationen der Klasse Has role mit Objekten der Klasse Role verbunden sein.

5.5 Modelltyp „Item Model“

Das Item Model beschreibt die beweglichen materiellen Teile im Supply Chain Network Modell. Diese dynamischen Objekte werden nicht direkt vom Modellierer modelliert, sondern während eines Simulationslaufs erzeugt und verbraucht. Da Teile nicht durch Instanziierung von Modellierungsklassen als Objekte modelliert werden, weisen sie im Supply Chain Networkmodell auch keine ‚realen‘[67] Attribute auf, mittels denen der Modellierer eine Verknüpfung zu Objekten des Item Models herstellen kann. Diese Zuweisung erfolgt durch Vergabe des Teiletyps in einem Einkaufs- oder Produktionsprogramm der Source- oder Make-Objekte.

Die Modellierung eines, auf dem Komponentenmodell basierenden Modells, ist keine Voraussetzung für die Anwendung der Methode SIMchronization. Gerade bei komplexeren Produktionsketten, kann aber eine Darstellung einer Stückliste oder einer baumartigen Struktur, welche die Entstehung des Endprodukts darstellt, sehr hilfreich für das Verständnis des Gesamtsystems sein.

Methodisch ist es sinnvoll, mit der Erstellung des Komponentenmodells zu beginnen und im nächsten Schritt für jeden Bearbeitungs- bzw. Transformationsschritt von Teilen einen Produktionsschritt im Supply Chain Network Modell zu erstellen[68].

---

[67] Im Rahmen eines Simulationslaufs verfügen die dynamischen Teile über ‚virtuelle‘ Attribute, auf die durch Regeln zugegriffen werden kann. Der Modellierer kann durch Formulierung entsprechender Regeln diese Attribute auslesen und belegen.

[68] Die Ableitung des groben Supply Chain Modells aus dem Item Model könnte auch unter Anwendung eines Algorithmus automatisiert umgesetzt werden. Jeder Bearbeitungs- oder Transformationsschritt im Item Model wird zu einem Make-Objekt im SCN-Modell. Produktionsprogramme der Make-Objekte können ebenfalls abgeleitet werden. Im Material flow channel zwischen den einzelnen Make-Objekten werden gemäß der Anzahl verschiedener Teiletypen der Produktionsprogramme angrenzender Make-Objekte von Teilen Store-Objekte platziert. Die weitere Ausgestaltung des Modells obliegt anschließend dem Modellierer.

Je nach Anwendungsdomäne der Methode sollte die Begrifflichkeit der verwendeten Klassen angepasst werden. Anhand einiger aus der Literatur bekannter Ontologien werden Beispiele dargestellt, wie Teile und die daraus resultierenden Produkte klassifiziert werden können. Aus einer, zur Domäne passenden Klassifikation kann ein entsprechendes Metamodell, beziehungsweise eine Erweiterung des von der Methode SIMchronization vorgegebenen Metamodells abgeleitet werden. Die Attributierung der Klassen hat keinen Einfluss auf die Funktionalität der Methode SIMchronization und dient nur der Dokumentation.

5.5.1 Onto-SCM

Die Onto-SCM (Ye et al., 2008b) wurde bereits im Kapitel 5.4.2 unter Ressourcen Gesichtspunkten betrachtet. Neben der SC-Resource Theory umfasst die Onto-SCM auch noch eine SC-Item Theory die sich mit Geschäftsobjekten befasst.

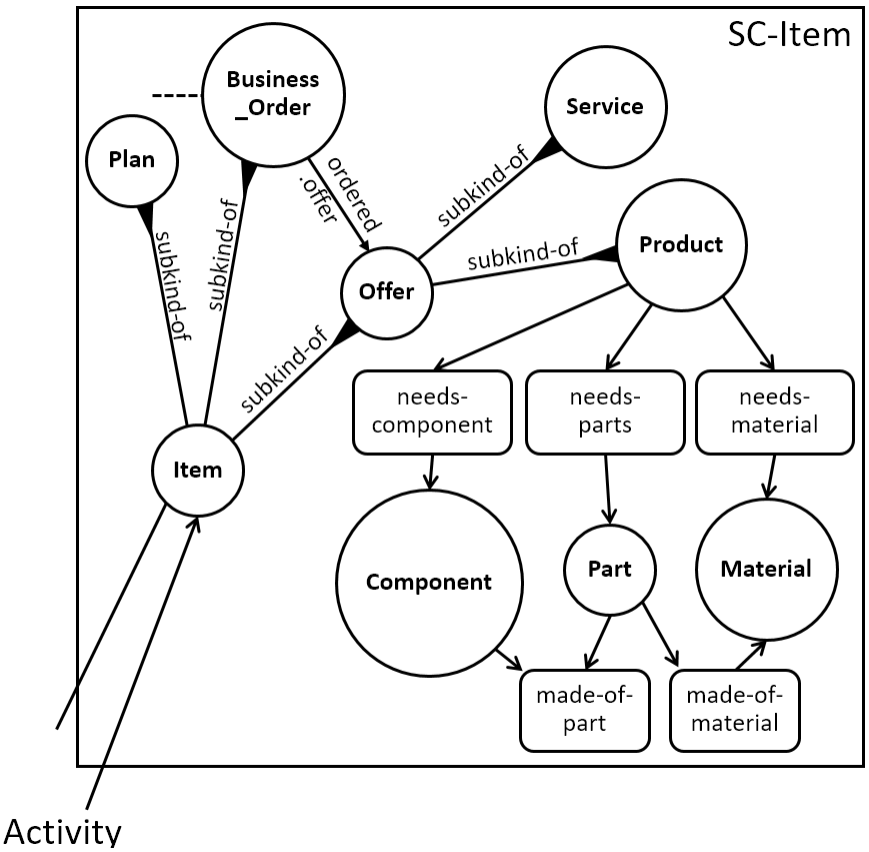


Abbildung 55: Ausschnitt SC-Item Theory der Onto-SCM (Ye et al., 2008b)

Ein Item der Onto-SCM kann sowohl materiell, wie der In-und Output eines Produktionsprozesses, als auch immateriell und abstrakt, wie ein Plan, eine Prognose oder ein Service, sein. Ein Item spezialisiert sich in Plan „Business_Order“ und Offer. Ein Offer steht dabei für Produkte und Services die ein Unternehmen dem Kunden anbietet. Die Klasse

Business_Order hingegen, stellt Aufträge, also Information dar, welche die Supply Chain steuern. Sie verfügt über die Subklassen Customer_Order, Purchase_Order and Sales_Order, welche in der Abbildung ausgeblendet sind. Ein Produkt besteht aus Komponenten, Parts und Material, wobei eine Komponente aus Parts und Parts wiederrum aus Material bestehen.

In der Methode SIMchronization, sind alle dargestellten Klassen, wenn auch in leicht unterschiedlicher Anordnung im Metamodell enthalten.

Der Klasse Plan der Onto-SCM entsprechen die steuernden Regeln in den Plan-Objekten und die auftragsbezogenen Regeln, die in den Objekten des Supply Chain Network Models hinterlegt sind. Die Business_Order finden ihre Abbildung in den, über Information flow channel versendeten Nachrichten und Abrufen. Die hier in der Onto-SCM verwendete Klasse Produkt kann mitsamt ihren Subklassen als temporäres Objekt „Teil“ aufgefasst werden.

5.5.2 AsD Ontology

K.-Y. Kim et al. (Kim et al., 2006) stellen mit der AsD Ontology (Assembly Design) eine formale Spezifikation vor, mit der Wissen zur Montage von Anlagen weitergegeben und auch von Maschinen interpretiert werden kann. Diese Ontologie ist im Gegensatz zu den bisher vorgestellten Klassifikationen nicht strategisch sondern operativ ausgerichtet. Demgemäß ist die Umsetzung domänenspezifisch und feingranular.

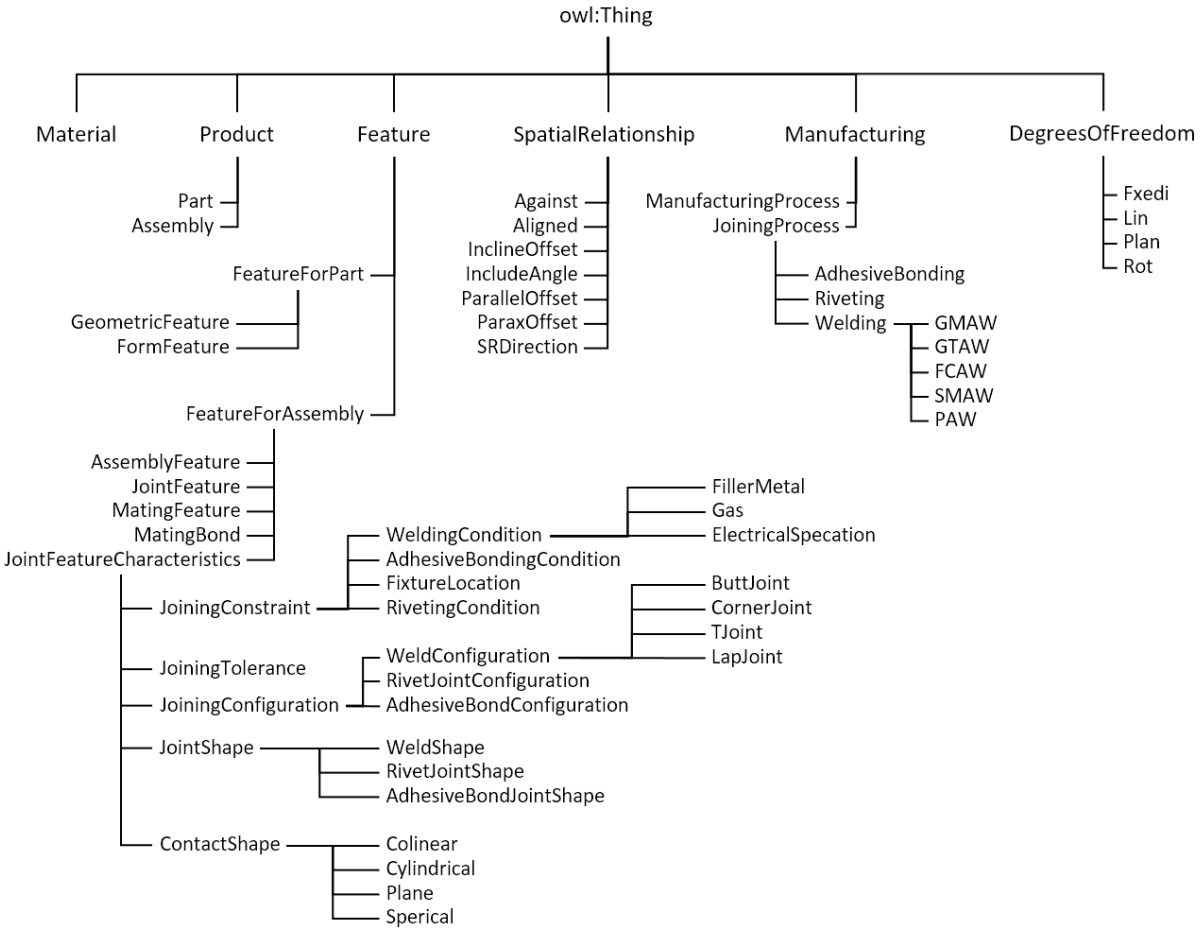


Abbildung 56: AsD Ontologie Klassenhierarchie (Kim et al., 2006)

Die Ontologie unterscheidet zwischen Materialien und Produkten, Produkte können wiederrum aus Teilen (Part) und Baugruppen (Assembly) bestehen. Eine Baugruppe besteht aus einer Gruppe von Teilen oder Komponenten die eigenständig Funktionen ausführen kann. Da die Ontologie ein übergreifendes Framework zum Design von Montagevorgängen für Anwender von kollaborativen Design Tools, wie Computer-Aided Design (CAD) Programmen, darstellt, werden die weiteren Klassen zur Beschreibung von Teilen, ihrer Eigenschaften, Montagevorgaben und Qualitätsanforderungen verwendet.

5.5.3 Metamodell des „Item Models“

Nach dem Studium einer strategisch (Onto-SCM) und einer operativ (AsD-Ontology) ausgerichteten Ontologie ergeben sich ähnliche Schlussfolgerungen wie bei der Betrachtung der Ontologien für das Ressourcenmodell. Da Modelle, die aus diesem Metamodell abgeleitet werden für die Anwendung der Methode SIMchronization ausschließlich dokumentarischen Charakter haben[69] werden auch hier die Klassen des Metamodells nur auf oberster Ebene

[69] Dies könnte sich ändern, wenn zukünftig ein automatisierter Abgleich der Supply Chain Network Model Struktur mit der Item Model Struktur erfolgt

vorgegeben. Eine Verfeinerung und klassenspezifische Attributierung kann durch den Anwender unter Bedachtnahme des Einsatzgebiets und Anwendungsziel vorgenommen werden.

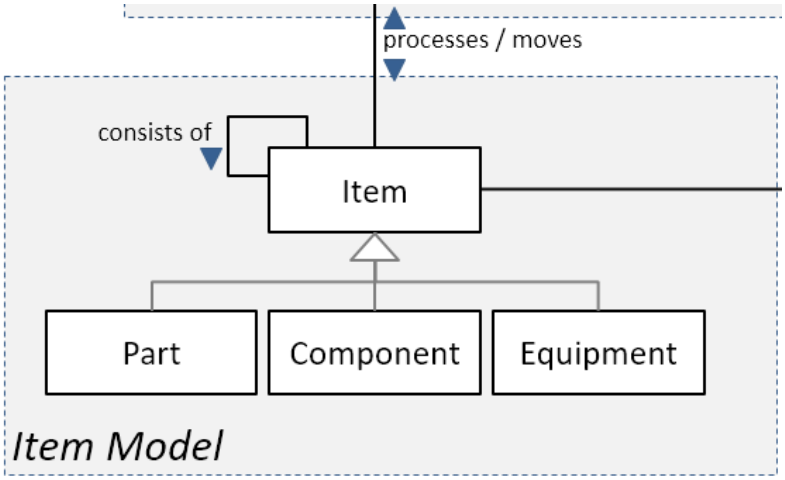


Abbildung 57: Metamodell des Item Modells der Methode SIMchronization

5.5.4 Modellierungsklasse „Item“

Ein Item oder Teil ist in der Methode SIMchronization jedes Stück eines Guts, das im Materialfluss bewegt und gegebenenfalls bearbeitet wird. Ein Teil kann aus weiteren Teilen bestehen. Diese Verbindung kann permanent, in Folge einer Verarbeitung von Teilen sein oder temporär, wenn Teile in einem anderen Teil verpackt werden.

Diese sehr offene Definition, ermöglicht es die Methode in allen Logistik relevanten Einsatzbereichen zu verwenden. Für die Domäne der Instandhaltung ist vor allem die Klassifikation in Ersatzteile, Bauteile und Anlagen wichtig. Item wird als Superklasse für die nachfolgenden Spezialisierungen definiert.

5.5.5 Modellierungsklasse „Part“

Das Teil, das in einer Baugruppe oder Anlage verbaut wird. Kann im Falle einer Instandsetzung ein Ersatzteil sein.

5.5.6 Modellierungsklasse „Component“

Eine in einer Anlage verbaute Komponente wird auch als Unit oder Baugruppe bezeichnet. Sie wird als eine Einheit gewartet und gegeben falls ausgetauscht.

5.5.7 Modellierungsklasse „Equipment“

Das Equipment oder die Anlage ist das instand zu haltende Objekt. Equipments können aus 1..n Components, Parts oder Items bestehen.

5.5.8 Beziehungstyp „Consists of“

Diese Beziehungsklasse kann dazu verwendet werden, um Items zu strukturieren und damit Stücklisten darzustellen.

5.6 Modelltyp „Information Model“

Die Kommunikation in der Supply Chain erfolgt auf Basis einer, allen betroffenen SC-Partnern bekannten, strukturellen beziehungsweise syntaktischen Vorgabe. Die semantische Bedeutung sowohl der Codes der Fragestellung als auch der Codes der Antworten sind bekannt. Die pragmatische Bedeutung der Information ergibt sich erst im konkreten Anwendungsfall und kann in der Methode SIMchronization mittels Regeln bestimmt werden.

Die Modellierung eines, auf dem Information Model basierenden Modells, ist keine Voraussetzung für die Anwendung der Methode SIMchronization. Es kann aber bei einem starken Kommunikationsbedarf in der Supply Chain von Vorteil sein, Standards für Nachrichten oder Abrufe im Modell zu erstellen und damit zu dokumentieren.

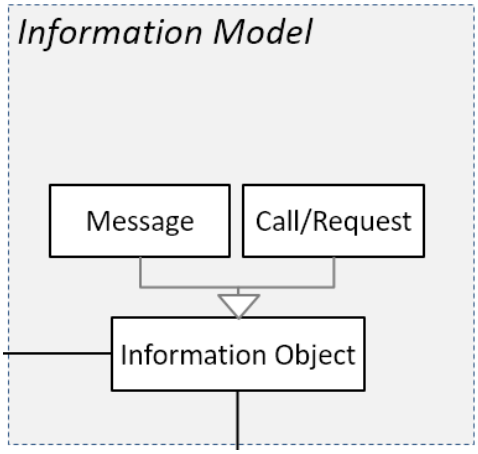


Abbildung 58: Metamodell des Information Model der Methode SIMchronization

Eine Information kann mittels zweier, alternativer Kommunikationsarten übermittelt, beziehungsweise erfragt werden.

5.6.1 Modellierungsklasse „Information Object“

Diese dynamischen Informationsobjekte werden nicht direkt vom Modellierer modelliert, sondern während eines Simulationslaufs erzeugt, versendet, empfangen und ausgewertet. Da Informationsobjekte nicht durch Instanziierung von Modellierungsklassen als Objekte modelliert werden, weisen sie im Supply Chain Network Model auch keine ‚realen'[70] Attribute auf, mittels denen der Modellierer eine Verknüpfung zu Objekten des Information Models herstellen kann. Diese Modellierung der Informationsobjekte und deren Spezialisierungen erfolgt ausschließlich zu Dokumentationszwecken.

5.6.2 Modellierungsklasse „Message“

Der Versand von Nachrichten stellt eine asynchrone Kommunikation dar. Abhängig vom Verhalten des Empfängers sendet dieser eine Antwort oder nicht. Eine eingehende Nachricht muss vom Empfänger nicht sofort bearbeitet werden, aber spätestens in der folgenden Periode. Der Zeitpunkt der Bearbeitung hängt von der Klasse und der Priorität des Empfängerobjekts und davon ab, ob das Objekt in dieser Periode schon bearbeitet wurde. Die Übermittlung einer Nachricht erfolgt immer genau an einen Empfänger, sie ist also Unicast (Ford und GmbH, 1998, S. 571).

5.6.3 Modellierungsklasse „Call / Request“

Ein Abruf entspricht einer synchronen Kommunikation zwischen Sender und Empfänger. Synchron bedeutet hierbei, dass auf die Anfrage eine sofortige Antwort gegeben oder vom Sender erwartet wird. Der Sender blockiert nach Absenden der Anfrage die weitere Bearbeitung und setzt diese erst fort, wenn die zugehörige Antwort eingetroffen ist. Der Abruf und die darauf folgenden Antwort wird immer an genau einen Empfänger gerichtet, er ist also Unicast (Ford und GmbH, 1998, S. 571).

[70] Im Rahmen eines Simulationslaufs verfügen die dynamischen Teile über ‚virtuelle' Attribute, auf die durch Regeln zugegriffen werden kann. Der Modellierer kann durch Formulierung entsprechender Regeln diese Attribute auslesen und belegen.

6. Beschreibung des Metamodells in FDMM

In diesem Kapitel wird das Metamodell, als Vorbereitung zur anschließenden Implementierung, formal exakt beschrieben. Basierend auf der plattform-unabhängigen Beschreibung der Klassen, Relationen und Attribute im vorherigen Kapitel 5 „Metamodell" detailliert die mathematische Formulierung bestimmte Aspekte des Metamodells, wie zum Beispiel Attributstypen, mögliche Beziehungen unter Objekten und deren Kardinalitäten. Die mathematische Formulierung kann einerseits dafür genutzt werden Kriterien für die Validität von Metamodellen und insbesondere den davon abgeleiteten Modellen zu definieren und andererseits zur Prüfung dieser Kriterien herangezogen werden.

Der von Fill et al. (Fill et al., 2012) vorgestellte Ansatz FDMM[71] ist speziell darauf ausgelegt, die Implementierung von Metamodellen auf der ADOxx Plattform zu unterstützen. Diese Eigenschaft ist auch der Hauptgrund, warum dieser Ansatz als formale Beschreibungssprache in diesem Dissertationsprojekt herangezogen wurde. Weitere Vorteile sind die relativ einfache Syntax, die kein höheres mathematisches Wissen erfordert. FDMM unterstützt auch die Beschreibung von Modellen, dieser Aspekt wurde in dieser Arbeit nicht verwendet und wird im Weiteren auch nicht erläutert.

Im Folgenden wird die Definition von Metamodellen mittels FDMM generell beschrieben. Dabei wird die Definitionsfolge von Fill (Fill et al., 2012) wiedergegeben und anschließend wird der Formalismus auf das in dieser Arbeit entwickelte Metamodell angewendet. Fill weist darauf hin, dass FDMM die Beschreibung der Kernkomponenten des ADOxx Ansatzes unterstützt, es handelt sich aber, zumindest in der vorliegenden Fassung, nicht um eine vollständige Beschreibungssprache aller Aspekte des ADOxx Metamodells.

6.1 Beschreibung des FDMM Formalismus

Nach Fill sind die Kernkomponenten eines Metamodells:

- die Modelltypen(im Folgenden als „MT" bezeichnet), jeweils bestehend aus
    - Objekttypen
    - Datentypen
    - Attributen
- die Vererbungsbeziehungen zwischen Objekttypen (im Folgenden als „$\preccurlyeq$" bezeichnet).
- die einem Objekttyp zugewiesenen Attribute (im Folgenden als „domain" bezeichnet)
- die Typisierung der Attribute (im Folgenden als „range" bezeichnet)
- die Kardinalität von Attributen (im Folgenden als „card" bezeichnet).

[71] FDMM steht für: „A Formalism for Describing ADOxx Meta Models and Models"

Ein Metamodell MM kann somit als Tupel der Form:

$$\mathrm{MM} = \langle\, \mathrm{MT}, \preccurlyeq, \mathrm{domain}, \mathrm{range}, \mathrm{card} \,\rangle$$

ausgedrückt werden, wobei MT die Menge aller Modelltypen des Metamodells ist

$$\mathrm{MT} = \{\, \mathrm{MT}_1, \mathrm{MT}_2, \mathrm{MT}_3, \mathrm{MT}_4, \ldots, \mathrm{MT}_m \,\}$$

Wie aus der obigen Auflistung zu ersehen kann ein spezifischer Modelltyp wieder als Tupel, bestehend aus einer Menge von Objekttypen O_i^T, einer Menge von Datentypen D_i^T und einer Menge von Attributen A_i dargestellt werden.

$$MT_i = \langle\, O_i^T, D_i^T, A_i \,\rangle$$

$$O^T = \bigcup_j O_j^T, D^T = \bigcup_i D_i^T, A = \bigcup_i A_i$$

ADOxx unterstützt verschiedene Objekttypen, sogenannte „Klassen“. Für die Realisierung des Metamodells in dieser Arbeit sind die Modellierungsklassen und die Beziehungsklassen wichtig, darüber hinaus verfügt ADOxx noch zumindest über die Objekttypen Tabellenklassen und Attributprofilklassen (BOC Information Technologies Consulting AG, 2014). Die ADOxx-Klassen und ihre Beziehung zueinander können in FDMM mit den oben vorgestellten Mitteln beschrieben werden, da Fill zur Beschreibung von möglichen Relationen zwischen Objekten auf die Verwendung von Attributen als Start und Endpunkt zurückgreift.

Um die Vererbungslogik von Objekttypen O^T des Metamodells auszudrücken, stellt der Formalismus eine Ordnungsfunktion $\preccurlyeq$ zur Verfügung. Dabei ist ein Objekttyp o_1^T ein Subtyp von o_2^T, wenn gilt $\mathrm{o}_1^T \preccurlyeq \mathrm{o}_2^T$. Der Subtyp erbt alle Attribute vom Supertyp, das bedeutet dass Attribute, die allen Objekttypen zugewiesen werden sollen, nur einmal in der Superklasse definiert werden müssen.

Um Attribute einzelnen Objekttypen zuzuordnen, wird die domain-Funktion eingeführt, sie bildet die Attribute auf die Potenzmenge aller Objekttypen ab.

$$domain\colon A \rightarrow \mathcal{P}\left(\bigcup_j \mathrm{O}_j^T\right)$$

Die Domain-Funktion beschränkt somit die Attribute, die bei der Instanziierung eines Objekts zur Verfügung stehen. Da einzelne Objekttypen sowohl Modellierungsklassen als auch Beziehungsklassen sein können, erlaubt die Funktion die Zuordnung von Attributen zu beiden Typen von Objekten.

Um den Typ der zugeordneten Attribute zu definieren, wird die range-Funktion eingeführt. Sie bildet ein Attribut auf die Potenzmenge aller Paare von Objekttypen und Modelltypen, alle Datentypen und alle Modelltypen ab.

$$range: A \rightarrow \mathcal{P}\left(\bigcup_j (O_j^T \times \{MT_j\}) \cup D^T \cup MT\right)$$

Offensichtlich dient das mapping von Attributen auf Datentypen dazu, den Attributtyp zu definieren. Ist zum Beispiel der Datentyp von Typ „Integer“ dann ist auch das entsprechende Attribut vom Typ „Integer“. ADOxx verfügt darüber hinaus über spezielle Attributtypen die die Referenzierung von Objekten oder Modellen ermöglichen. Die Paarbildung von Objekttypen und Modelltypen erlaubt somit eine Beschränkung der zu referenzierenden Objekte auf Objekte eines bestimmten Objekttyps, die in einem Modell eines bestimmten Modelltypen instantiert wurden. Soll ein Objekt ein ganzes Modell referenzieren, so kann mittels Angabe eines oder mehrere Modelltypen die Zuordnungsmöglichkeiten beschränkt werden.

Um die Anzahl der Werte, die ein Attribut enthalten kann, einzuschränken, wird die card-Funktion eingeführt. Sie bildet Paare von Objekttypen und Attributen auf Paare von nicht-negativen ganzen Zahlen (inklusive ∞) ab.

$$card: O^T \times A \rightarrow \mathcal{P}\big(\mathbb{N} \times (\mathbb{N} \cup \{\infty\})\big)$$

Die Funktion erlaubt beispielsweise die Anzahl der auswählbaren Werte in einem List-Box Attribut zu beschränken. Ist der Minimalwert 1 und der Maximalwert 1, so muss genau ein Wert gewählt werden. Ist der Minimalwert 0 und der Maximalwert ∞, so kann eine beliebige Anzahl von Werten in diesem Attribut gewählt werden. FDMM nutzt diese Funktion auch um die Anzahl der möglichen Beziehungen zwischen Objekten zu limitieren.

Die drei oben angeführten Funktionen können nun herangezogen werden, um Beziehungen im ADOxx Metamodell zu definieren, die bei der Instanziierung zwischen Objekten oder zwischen Objekten und Modellen gebildet werden können. Um zu ermöglichen, dass Beziehungen desselben Beziehungstyps von Objekten unterschiedlicher Objekttypen starten

oder enden können, können diese Objekttypen unter einer Superklasse zusammengefasst werden. Formal kann dies mittels der oben beschriebenen Ordnungsfunktion $\preccurlyeq$ definiert werden. Beziehungen im Metamodell können gerichtete oder ungerichtet definiert werden. Beziehungen in ADOxx werden immer gerichtet angelegt[72] und müssen über einen Start- und Endpunkt verfügen. Sie können als Beziehungsklasse oder Referenz definiert werden.

Beziehungsklasse: Dieser Typ von Beziehung wird auch Relation genannt und ist ein eigenständiges Objekt, das ein Start- und Endobjekt vom Typ Modellierungsklasse benötigt, um zu existieren. Graphisch werden Relationen im Modell als Kante abgebildet, häufig als Pfeil. Da jede Relation ein eigenständiges Objekt ist, können dem Beziehungstyp Attribute zugewiesen werden. Auf Metamodellebene können sich Objekte mittels einer Beziehung selbst referenzieren, eine Unterscheidung zwischen Start- und Endpunkt der Beziehung im Metamodell ist in diesem Fall eigentlich nicht notwendig. In ADOxx ist es allerdings nicht möglich, dass eine Beziehung an derselben Objektinstanz startet wie endet. Eine Selbstreferenz im Metamodell drückt daher aus, dass die Modellierung einer Sequenz von Objekten desselben Objekttyps[73] möglich ist. Referenzen sind immer gerichtet und müssen über einen definierten Start- und Endpunkt verfügen. Eine Einschränkung der für diese Arbeit herangezogenen Version von ADOxx ist, dass nur maximal eine Beziehung einer Beziehungsklasse zwischen denselben Start- und Endobjekten angelegt werden kann (1:1-Relation).

Referenzen: Referenzen können in ADOxx zwischen Objekten untereinander und zwischen Objekten und Modellen gebildet werden. Auf Instanziierungsebene erfolgt dies durch Eintrag eines eindeutigen Objekt- oder Modellidentifizierers zur Definition des Endpunktes im entsprechenden Attribut des Startobjekts. Da ADOxx für Referenzattribute die Angabe multipler Zielobjekte erlaubt, können 1:n-Relationen dargestellt werden. Die Angabe spezifischer Attribute für eine Referenz ist nicht möglich.

FDMM unterstützt die Beschreibung beider Beziehungstypen im Metamodell. Um die Beschreibung von Beziehungsklassen zu ermöglichen, die ja als eigenes Objekt definiert werden, behilft sich der Formalismus mit der Definition von zwei Attributen, von denen eines das oder die möglichen Start-Objekttypen und das anderer das oder die möglichen Ende-

[72] Auch wenn in der graphischen Darstellung eines Modells durch Weglassen eines Richtungsindikators, wie zum Beispiel einer Pfeilspitze, eine ungerichtete Beziehung semantisch ausgedrückt wird, verfügt dieses Beziehungsobjekt über einen definierten Start- und Endpunkt und ist deshalb technisch gesehen gerichtet.

[73] Ein typisches Beispiel hierfür sind Objekte von Objekttyp „Task“ die mittels einer Beziehung „Nachfolger“ eine Sequenz und damit einen Prozessfluss bilden.

Objekttypen beschreibt. Die Definition von Attributen dieser Beziehungsklasse entspricht der Attributdefinition von Modellierungsklassen.

Die Beschreibung von Referenzen erfolgt auf die gleiche Weise wie für andere Attribute eines Modellierungsobjekts. Mittels der domain-Funktion wird der Objekttyp angegeben, dem das Attribut zugewiesen wird, die range-Funktion definiert mögliche Ziel-Objekttypen oder Ziel-Modelltypen und die card-Funktion gibt an wie viele Referenzen ausgehend von einem Attribut/Objekt erlaubt sind.

Fill definiert in seiner Arbeit noch einige Korrektheitskriterien, wie die Forderung, dass die Mengen von Objekttypen, Datentypen und Attributen paarweise disjunkt sein müssen oder dass die Zuweisung von Attributen, Objekttypen und Modelltypen über alle Formeln hinweg konsistent sein muss.

6.2 Anwendung von FDMM zur Beschreibung des Metamodells

Der dargestellte Formalismus wird nun dafür hergezogen, um das in Kapitel 5 beschriebene Metamodell formal genauer zu beschreiben. Dies ist der letzte Schritt bevor das Metamodell prototypisch auf der Metamodellierungsplattform ADOxx implementiert wird.

Der Name des Metamodells ist „SIMchronization“

$$MM: SIMchronization$$

Es setzt sich zusammen aus vier Modelltypen, der Ordnungsbeziehung, und den Attributdefinitionen mittels domain, range und card Funktionen.

6.2.1 Modelltypen, Objekttypen und deren Vererbung

Das Metamodell besteht aus vier Modelltypen:

$$MT_{SCNM}: Supply\ Chain\ Network\ Model$$

$$MT_{ITM}: Item\ Model$$

$$MT_{RM}: Resource\ Model$$

$$MT_{INM}: Information\ Model$$

Die Objekttypen des Supply Chain Network Modells sind:

$$O^{T}_{SCNM} = \{\ \textit{SCNM-Classes},\ \textit{Plan},\ \textit{Material flow object},\ \textit{Make},\ \textit{Source},\ \textit{Store},\ \textit{Deliver},\ \textit{Transport},\ \textit{Switch},\ \textit{Reader},\ \textit{Material flow channel},\ \textit{Information flow channel}\ \}$$

Die Ordnungsfunktion definiert die Superklasse *SCNM-Classes*:

$$\textit{Material flow object} \preccurlyeq \textit{SCNM-Classes},\quad \textit{Plan} \preccurlyeq \textit{SCNM-Classes}$$

Die Ordnungsfunktion definiert die Superklasse *Material flow object*:

$$\textit{Make} \preccurlyeq \textit{Material flow object},\ \textit{Source} \preccurlyeq \textit{Material flow object},\ \textit{Store} \preccurlyeq \textit{Material flow object},\ \textit{Deliver} \preccurlyeq \textit{Material flow object},\ \textit{Transport} \preccurlyeq \textit{Material flow object},\ \textit{Switch} \preccurlyeq \textit{Material flow object},\ \textit{Reader} \preccurlyeq \textit{Material flow object}$$

Die Objekttypen des Item Model sind:

$$O^{T}_{ITM} = \{\ \textit{Item},\ \textit{Part},\ \textit{Component},\ \textit{Equipment},\ \textit{Consists of}\ \}$$

Die Ordnungsfunktion definiert die Superklasse *Item*:

$$\textit{Part} \preccurlyeq \textit{Item},\ \textit{Component} \preccurlyeq \textit{Item},\ \textit{Equipment} \preccurlyeq \textit{Item}$$

Die Objekttypen des Resource Model sind:

$$O_{RM}^{T} = \{ \begin{array}{l} \textit{Resource}, \\ \textit{Tool}, \\ \textit{Actor}, \\ \textit{Role}, \\ \textit{IT-System}, \\ \textit{Has role} \\ \} \end{array}$$

Die Ordnungsfunktion definiert die Superklasse *Resource*:

$$\textit{Tool} \preccurlyeq \textit{Resource}, \textit{Actor} \preccurlyeq \textit{Resource}, \textit{Role} \preccurlyeq \textit{Resource}, \textit{IT-System} \preccurlyeq \textit{Resource}$$

Die Objekttypen des *Information Model* sind:

$$O_{INM}^{T} = \{ \begin{array}{l} \textit{Information object}, \\ \textit{Message}, \\ \textit{Call / Request} \\ \} \end{array}$$

Die Ordnungsfunktion definiert die Superklasse *Information object*:

$$\textit{Message} \preccurlyeq \textit{Information object}, \textit{Call / Request} \preccurlyeq \textit{Information object}$$

6.2.2 Attribute der Objekttypen

Im Folgenden werden die Attribute der Objekttypen formal beschrieben. Die fachliche Beschreibung der einzelnen Attribute wird in Kapitel 5 „Metamodell“ behandelt.

6.2.3 Verwendete Datentypen

Die range-Funktion weist den Attributen einen Datentyp zu. Da FDMM keine Datentypen zur Verwendung vorschreibt, werden die im Folgenden verwendeten Datentypen kurz erläutert. Sie basieren auf Datentypen der Plattform ADOxx (BOC Information Technologies Consulting AG, 2014):

SingleLineString: Bis zu 250 Zeichen ohne Zeilenumbruch. Defaultwert ist der Leerstring “”.

ShortText: Bis zu 3.700 beliebiger Zeichen, inklusive Zeilenumbruch. Defaultwert ist der Leerstring “”.

LongText: Bis zu 32.000 beliebiger Zeichen, inklusive Zeilenumbruch. Defaultwert ist der Leerstring "".

NichtNegativeGanzzahl: Nicht negative ganze Zahlen. Defaultwert ist 0.

Time: Beschreibt eine Zeitdauer oder ein Datum im Format Jahre:Tage:Stunden:Minuten:Sekunden Defaultwert ist „00:000:00:00:00"

Referenz: Ein Sonderfall ist der Datentyp Referenz, da FDMMs range-Funktion dem Attribut nicht den eigentlichen Datentyp zuweist, sondern den Objekt- oder Modelltyp des Zielobjekts.

Gleitkommazahl: positive oder negative Zahl bestehend aus maximal 16 Ziffern inklusive maximal 6 Nachkommastellen. Defaultwert ist 0,00.

Alle Modellierungsobjekttypen benötigen die Attribute „Name" und „Description". Diese Attribute werden deshalb den Superklassen *SCNM-Classes, Item, Resource* und *Information object* zugewiesen und damit an deren Subklassen vererbt.

domain (Name) = { SCNM-Classes, Item, Resource, Information object }
range (Name) = { SingleLineString }
card (Name) = (1, 1)

domain (Description) = { SCNM-Classes, Item, Resource, *Information object }*
range (Description) = { LongText }
card (Description) = (1, 1)

6.2.4 Attribute der Objekttypen des Modelltyps „Supply Chain Network Modell"

6.2.4.1 Attribute des Objekttyps „Make"

Das Attribut „Production program" nimmt einen Regelausdruck auf, der das Produktionsprogramm des Make Objekts beschreibt. Die Syntax dieses Ausdrucks wird in Kapitel 7.5.1.1 beschrieben.

domain (Production program) = { Make }
range (Production program) = { ShortText }
card (Make, Production program) = (1, 1)

domain (Order) = { Make }
range (Order) = { Time }
card (Make, Order) = ⟨ 1, 1 ⟩

domain (Remaining production time) = { Make }
range (Remaining production time) = { Time }
card (Make, Remaining production time) = ⟨ 1, 1 ⟩

domain (Production history) = { Make }
range (Production history) = { LongText }
card (Make, Production history) = ⟨ 1, 1 ⟩

Mittels eines Regelausdrucks im Attribut "Fixed costs per Order" und/oder „Execution time variable costs" können die entstehenden Kosten pro Produktionsgang beschrieben werden.

domain (Fixed costs per Order) = { Make }
range (Fixed costs per Order) = { ShortText }
card (Fixed costs per Order) = ⟨ 1, 1 ⟩

domain (Execution time variable costs) = { Make }
range (Execution time variable costs) = { ShortText }
card (Execution time variable costs) = ⟨ 1, 1 ⟩

Beim Attribut „Resource" handelt es sich um den ein Referenzattribut. Es können von einem Make-Objekt beliebig viele Resources, beziehungsweise deren Subtypen referenziert werden. Auf die Ausformulierung der Resource-Attribute der restlichen Objekttypen wird verzichtet.

domain (Resource) = { Make }
range (Resource) = { Resource}
card (Make , Resource) = ⟨ 0, ∞ ⟩

6.2.4.2 Attribute des Objekttyps „Plan"

Das Attribut "Timestamp" enthält die aktuelle Zeit

domain (Timestamp) = { Plan }
range (Timestamp) = { Time }
card (Plan, Timestamp) = ⟨ 1, 1 ⟩

Das Attribut "Micro step" bildet die Sequenz der Aktionen im Modell zu einem Zeitpunkt ab.

domain (Micro step) = { Plan }
range (Micro step) = { NichtNegativeGanzzahl }
card (Plan, Micro step) = (1, 1)

Aktionen die das Plan-Objekt setzt können mittels eines Regelausdrucks im Attribut „Action" definiert werden.

domain (Action) = { Plan }
range (Action) = { ShortText }
card (Plan, Action) = (1, 1)

Reaktionen die das Plan-Objekt ausführt können mittels eines Regelausdrucks im Attribut „Reaction" definiert werden.

domain (Reaction) = { Plan }
range (Reaction) = { ShortText }
card (Plan, Reaction) = (1, 1)

Der Textinhalt des Attributs "Display" wird am Plan-Objekt visualisiert.

domain (Display) = { Plan }
range (Display) = { ShortText }
card (Plan, Display) = (1, 1)

6.2.4.3 Attribute des Objekttyps „Source"

Der Regelausdruck im Attribut „Order" wird während der Simulation ausgewertet und ergibt eine nicht-negative ganze Zahl, das Auftragsvolumen.

domain (Order) = { Source }
range (Order) = { ShortText }
card (Source, Order) = (1, 1)

Das Attribut „Sourcing program" nimmt einen Regelausdruck auf, der das Einkaufsprogramm des Source Objekts beschreibt. Die Syntax dieses Ausdrucks wird in Kapitel 7.5.2 beschrieben.

domain (Sourcing program) = { Source }
range (Sourcing program) = { ShortText }
card (Source, Sourcing program) = (1, 1)

domain (Order history) = { Source }
range (Order history) = { LongText }
card (Source, Order history) = (1, 1)

Mittels eines Regelausdrucks im Attribut "Cost function" können die entstehenden Kosten pro Einkaufsvorgang beschrieben werden.

domain (Cost function) = { Source }
range (Cost function) = { ShortText }
card (Source, Cost function) = (1, 1)

6.2.4.4 Attribute des Objekttyps „Store"

Das Attribut „Item type" ist ein Referenzobjekt auf genau ein Objekt des Objekttyps „Item" oder eines Subtyps.

domain (Item type) = { Store }
range (Item type) = { Item }
card (Store, Item type) = (1, 1)

Der aktuelle Lagerstand wird im Attribut „Stock level" angegeben.

domain (Stock level) = { Store}
range (Stock level) = { NichtNegativeGanzzahl }
card (Store, Stock level) = (1, 1)

Die Auflistung der Objekte vom Typ Teil oder eines Subtyps wird im Attribut " Items in stock" beziehungsweise „Items waiting to be stored" vorgehalten.

domain (Items in stock) = {Store}
range (Items in stock) = { LongText }
card (Store, Items in stock) = (1, 1)

domain (Items waiting to be stored) = {Store}
range (Items waiting to be stored) = { LongText }
card (Store, Items waiting to be stored) = (1, 1)

domain (Storage history) = {Store}
range (Storage history) = { LongText }
card (Store, Storage history) = (1,1)

Die Zustandsveränderung eines Teils während der Lagerung kann in den folgenden Attributen definiert und nachvollzogen werden.

domain (Condition adjustment) = { Store }
range (Condition adjustment) = { Regelausdruck }
card (Store, Condition adjustment) = (1, 1)

domain (Condition history) = { Store }
range (Condition history) = { LongText }
card (Store, Condition history) = (1, 1)

Mittels eines Regelausdrucks in den Attributen „Handling cost per item“ und „Storage cost per period and item“ können die entstehenden Lagerkosten beschrieben werden. „Handling cost per item“ kann die entstehenden Ein-/Auslagerungskosten pro Teil beschreiben und „Storage cost per period and item“ eine Funktion zur Beschreibung der variablen Lagerkosten enthalten.

domain (Handling cost per item) = { Store }
range (Handling cost per item) = { ShortText }
card (Store, Handling cost per item) = (1, 1)

domain (Storage cost per period and item) = { Store }
range (Storage cost per period and item) = { ShortText }
card (Store, Storage cost per period and item) = (1, 1)

6.2.4.5 Attribute des Objekttyps „Deliver“

Der Regelausdruck im Attribut „Order“ wird während der Simulation ausgewertet und ergibt eine nicht-negative ganze Zahl, das Liefervolumen.

domain (Order) = { Deliver }
range (Order) = { ShortText }
card (Deliver, Order) = (1, 1)

Das Attribut „Delivery program“ nimmt einen Regelausdruck auf, der das Lieferprogramm des Deliver-Objekts beschreibt. Die Syntax dieses Ausdrucks wird in Kapitel 7.5.3 beschrieben.

domain (Delivery program) = { Deliver }
range (Delivery program) = { ShortText }
card (Source, Delivery program) = (1, 1)

domain (Delivery history – to be) = { Deliver }
range (Delivery history – to be) = { LongText }
card (Source, Delivery history – to be) = (1, 1)

domain (Delivery history – as is) = { Deliver }
range (Delivery history – as is) = { LongText }
card (Source, Delivery history – as is) = (1, 1)

Mittels eines Regelausdrucks im Attribut Revenue function" können die entstehenden Erlöse pro Verkaufsvorgang beschrieben werden.

domain (Revenue function) = { Deliver }
range (Revenue function) = { Regelausdruck }
card (Deliver, Revenue function) = (1, 1)

6.2.4.6 Attribute des Objekttyps „Transport"

Das Attribut „Item type" ist ein Referenzobjekt auf genau ein Objekt des Objekttyps Item oder eines Subtyps.

domain (Item type) = { Transport }
range (Item type) = { Item }
card (Transport, Item type) = (1, 1)

Die aktuelle Anzahl an am Bestimmungsort angekommenen Teilen wird im Attribut „Number of items arrived" angegeben.

domain (Number of items arrived) = { Transport }
range (Number of items arrived) = { NichtNegativeGanzzahl }
card (Transport, Number of items arrived) = (1, 1)

domain (Transport time) = { Transport }
range (Transport time) = { Time }
card (Transport, Transport time) = (1, 1)

Die Auflistung der Objekte vom Typ Teil oder eines Subtyps die sich zur Zeit im Transport befinden oder welche gerade entladen werden wird in den folgenden Attributen vorgehalten.

domain (Items in transport) = { Transport }
range (Items in transport) = { LongText }
card (Transport, Items in transport) = (1, 1)

domain (Items arrived) = { Transport }
range (Items arrived) = { LongText }
card (Transport, Items arrived) = (1, 1)

Die Zustandsveränderung eines Teils während des Transports kann in den folgenden Attributen definiert und nachvollzogen werden.

domain (Condition adjustment) = { Transport }
range (Condition adjustment) = { Regelausdruck }
card (Transport, Condition adjustment) = (1, 1)

domain (Condition history) = { Transport }
range (Condition history) = { LongText }
card (Transport, Condition history) = (1, 1)

Mittels eines Regelausdrucks im Attribut „Transport cost per item" können die entstehenden Transportkosten beschrieben werden. Dabei können Ver- und Entladekosten pro Teil und Transportkosten pro Periode und Teil berücksichtigt werden.

domain (Transport cost per item) = { Transport }
range (Transport cost per item) = { ShortText }
card (Transport, Transport cost per item) = (1, 1)

6.2.4.7 Attribute des Objekttyps „Switch"

Dem Objekttyp Switch werden keine weiteren Attribute zugewiesen.

6.2.4.8 Attribute des Objekttyps „Reader"

Im Attribut „Stack" werden alle Objekte vom Typ Item oder eines seiner Subtypen aufgelistet, die das Objekt vom Typ Reader passieren.

domain (Stack) = { Reader }
range (Stack) = { LongText }
card (Reader, Stack) = (1,1)

Aktionen die das Reader-Objekt setzt, können mittels eines Regelausdrucks im Attribut „Action" definiert werden.

domain (Action) = { Reader }
range (Action) = { ShortText }
card (Reader, Action) = (1, 1)

Reaktionen die das Reader-Objekt ausführt, können mittels eines Regelausdrucks im Attribut „Reaction" definiert werden.

domain (Reaction) = { Reader }
range (Reaction) = { ShortText }
card (Reader, Reaction) = (1, 1)

6.2.4.9 Attribute des Objekttyps „Material flow channel"

Der Objekttyp „Material flow channel" ist eine Beziehungsklasse. FDMM sieht vor, jeweils ein Attribut zur Definition des Start- und Endpunkts zu definieren, das die möglichen Objekte über deren Typisierung beschränkt. Die Beziehung Materialfluss soll von und zu Subtypen der Superklasse „Material flow object" geführt werden können.

domain (Material flow channel-Start) = { Material flow channel }
range (Material flow channel-Start) = { Material flow object }
card (Material flow channel, Material flow channel-Start) = (1, 1)

domain (Material flow channel-Ende) = { Material flow channel }
range (Material flow channel-Ende) = { Material flow object }
card (Material flow channel, Material flow channel-Ende) = (1, 1)

Ein Attribut "Index" wird benötigt um die alternativen Materialflüsse nach einem Switch-Objekt anzusprechen

domain (Index) = { Material flow channel }
range (Index) = { NichtNegativeGanzzahl }
card (Material flow channel, Index) = (1, 1)

6.2.4.10 Attribute des Objekttyps „Information flow channel"

Der Objekttyp Information flow channel ist eine Beziehungsklasse. Die Beziehung Informationsfluss soll von und zu Subtypen der Superklasse SCMN-Classes geführt werden können.

domain (Information flow channel-Start) = { Information flow channel }
range (Information flow channel-Start) = { SCMN-Classes }
card (Information flow channel, Information flow channel-Start) = (1, 1)

domain (Information flow channel-Ende) = { Information flow channel }
range (Information flow channel-Ende) = { SCMN-Classes }
card (Information flow channel, Information flow channel-Ende) = (1, 1)

Ein Attribut "Index" wird benötigt, um die alternativen Informationsflüsse in Regeln ansprechen zu können.

domain (Index) = { Information flow channel }
range (Index) = { NichtNegativeGanzzahl }
card (Information flow channel, Index) = (1, 1)

6.2.4.11 Attribute des Objekttyps „Information object"

Der Objekttyp *Information object* und seine Spezialisierungen Message und Call/Request verfügen über keine Attribute.

6.2.5 Attribute der Objekttypen des Modelltyps „Item Model"

6.2.5.1 Attribute des Objekttyps „Item"

Da Item als Superklasse definiert wurde, vererben sich die Attribute auch an die Klassen Part, Component und Equipment.

domain (Typ) = { Item }
range (Typ) = { ShortText }
card (Item, Typ) = (1, 1)

domain (Unit costs) = { Item }
range (Unit costs) = { Gleitkommazahl }
card (Item, Unit costs) = (1, 1)

6.2.5.2 Definition des Beziehungstyps „Consists of"

Mittels der Relation Consists of können Stücklisten aufgebaut werden.

domain (Consists of-Start) = { Consists of }
range (Consists of-Start) = { Item }
card (Consists of, Consists of-Start) = (1, 1)

domain (Consists of-Ende) = { Consists of }
range (Consists of-Ende) = { Item }
card (Consists of, Consists of-Ende) = (1, 1)

6.2.6 Attribute der Objekttypen des Modelltyps „Resource Model“

6.2.6.1 Attribute des Objekttyps „Resource“

Da Resource als Superklasse definiert wurde, vererben sich die Attribute auch an Tool, Role, Actor und IT-System.

domain (Fix cost) = { Resource }
range (Fix cost) = { Gleitkommazahl }
card (Resource, Fix cost) = (1, 1)

domain (Variable cost) = { Resource }
range (Variable cost) = { Gleitkommazahl }
card (Resource, Variable cost) = (1, 1)

6.2.6.2 Definition des Beziehungstyps „Has role“

Mittels der Relation Has role können Akteuren Rollen zugewiesen werden.

domain (Has role-Start) = { Has role }
range (Has role-Start) = { Actor }
card (Has role, Has role-Start) = (0, 1)

domain (Has role-Ende) = { Has role }
range (Has role -Ende) = { Role }
card (Has role, Has role-Ende) = (0, 1)

6.2.7 Potentielle Erweiterung des Metamodells

Eine offensichtliche Erweiterung des Metamodells wäre eine Verlinkungsmöglichkeit von Modellen. Zum einen horizontale Verlinkung von Modellen desselben Modelltyps, um eine durchgängigen Wertschöpfungskette, die sich aus mehreren, typischerweise sequentiell angeordneten, Supply Chain Network Modellen zusammensetzt oder einer vertikalen Verlinkung, um ein Element eines statischen Supply Chain Objektes in einer feineren Detaillierung graphisch zu beschreiben.

Im Folgenden wird anhand des Objekttyps „Make“ eine derartige Verlinkung beispielhaft in FDMM beschrieben:

domain (Make-Referenz) = { Make }
range (Make-Referenz) = { MT_{SCNM} }
card (Make, Make-Referenz) = ⟨0,1⟩

Ist der Detailierungsgrad des instanziierten Objekts vom Objekttyp Make ausreichend, so muss keine Referenz auf ein Modell vom Modelltyp Supply Chain Network Model angelegt werden.

Offensichtlich muss im darunterliegenden Modell der dazugehörige Startpunkt des Materialflusses definiert werden, um einen durchgängige Passage der dynamischen Objekte des Materialflusses zu ermöglichen. Um Informationen von und zu statischen Objekten in anderen Supply Chain Network Modellen zu übertragen, muss ein Konzept eines modellübergreifenden Informationsflusses entwickelt werden.

6.2.8 Diskussion des Kapitels

Auch wenn einige der in diesem Kapitel abgebildeten Zusammenhänge bereits aus vorherigen Kapitel bekannt sind und dort ausreichend detailliert wurden, brachte der formale Ansatz dem Autor doch mehr Klarheit über den Aufbau des Metamodells und führte zu einigen Adaptionen des Metamodells. Dahingehend wurde ein Ziel von FDMM erfüllt.

7. Implementierung

Nachdem in der Design und Formalization Phase das Metamodell definiert und die zentralen Anforderungen an Algorithmen im konzeptionellen Modell grob spezifiziert wurden, folgt als nächster Schritt gemäß dem OMiLAB Life Cycle (siehe Kapitel 2.2.3) die Phase der eigentliche Implementierung. Es wird das zunächst plattformunabhängige Metamodell in ein plattformspezifisches Metamodell übertragen und implementiert. Wie schon ausgeführt, wird als Implementierungsplattform das Metamodellierungstool ADOxx verwendet.

Abbildung 59: ADOxx Startup-Screen

7.1 Umsetzung des Konzeptionellen Modells

Das konzeptionelle Modell der Methode SIMchronization sieht die Verwendung unterschiedlicher, spezialisierter Tools zur Umsetzung der Komponenten Modellierung, Simulation, Regel-Engine und Animation vor. Nach dem Studium der infrage kommenden Technologien und Abschätzung des Implementierungsaufwands für Schnittstellen wurde entschieden, dass die gesamte Funktionalität auf der Plattform ADOxx umgesetzt wird. Dies hat den Vorteil, dass die Daten in einem System gehalten werden und die Benutzerinteraktion in einer Oberfläche erfolgen kann. Es tritt auch kein Verlust an Information oder Semantik an den Schnittstellen auf. Des Weiteren verfügt der Autor über langjährige Erfahrung in der Umsetzung von Modellierungssprachen inklusive Programmierung zusätzlicher Funktionalitäten auf der Plattform ADOxx. Es ist somit zu erwarten, dass mittels des erstellten Prototyps gezeigt werden kann, dass die Methode SIMchronization, die in Kapitel 3 aufgestellten Anforderungen erfüllen kann.

7.2 Implementierung des Metamodells

ADOxx stellt das in Abbildung 60 dargestellte Meta²-Modell zur Verfügung. Die darin verfügbaren Konstrukte können vom Metamodellierer nicht verändert werden, sondern werden dafür genutzt, um Klassen des domänenspezifischen Metamodells davon abzuleiten.

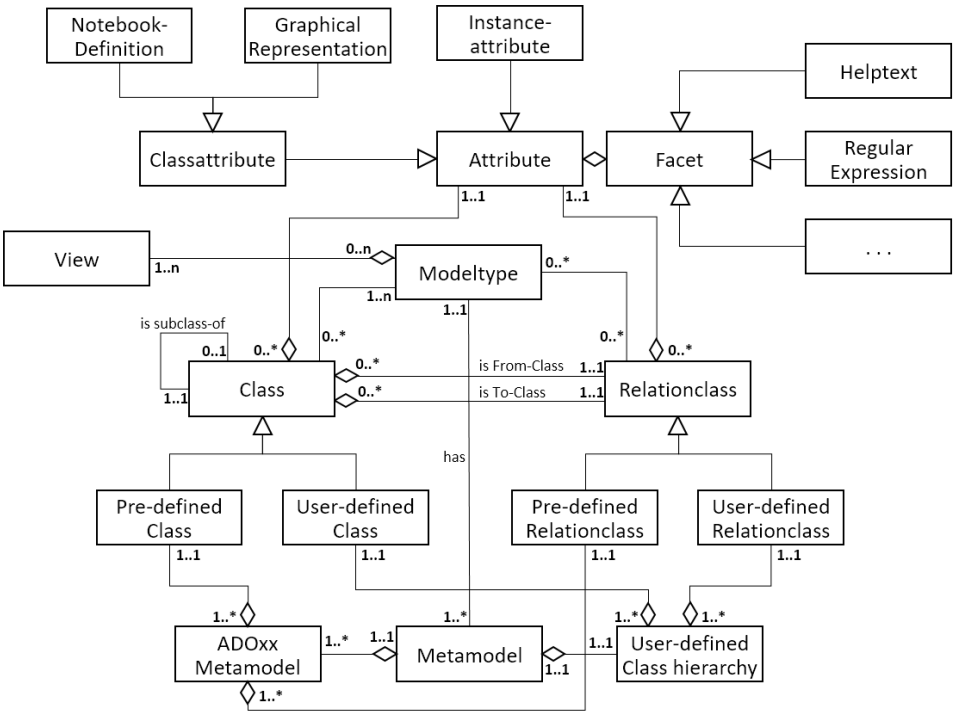


Abbildung 60: Das ADOxx Meta²-Modell (Fill und Karagiannis, 2013)

Dieses Meta²-Modell bildet die Struktur für die folgende Beschreibung der Implementierung des domänenspezifischen Metamodells der Methode SIMchronization.

7.2.1 Klassen und Relationen

Gemäß der Spezifikation des Metamodells werden zunächst die Klassen zur Modellierung der statischen Objekte vom Meta²-Modell abgeleitet. Dabei werden domänenspezifische, benutzerdefinierte Klassen erstellt und bereits in ADOxx vordefinierte Klassen dem Metamodell der Methode SIMchronization hinzugefügt. Alle Klassen des Meta²-Modells in ADOxx sind von der abstrakten Super-Klasse __D-construct__ abgeleitet. Relationen, auch als Beziehungsklassen bezeichnet, werden zwischen Modellierungsklassen angelegt.

Die Modelltypen werden in ADOxx auf Basis der erstellten Klassen definiert, deshalb wird zunächst das implementierte Klassenmodell vorgestellt. Abbildung 61 zeigt das Klassenmodell des Supply Chain Network Models mit seinen Modellierungsklassen und Relationen.

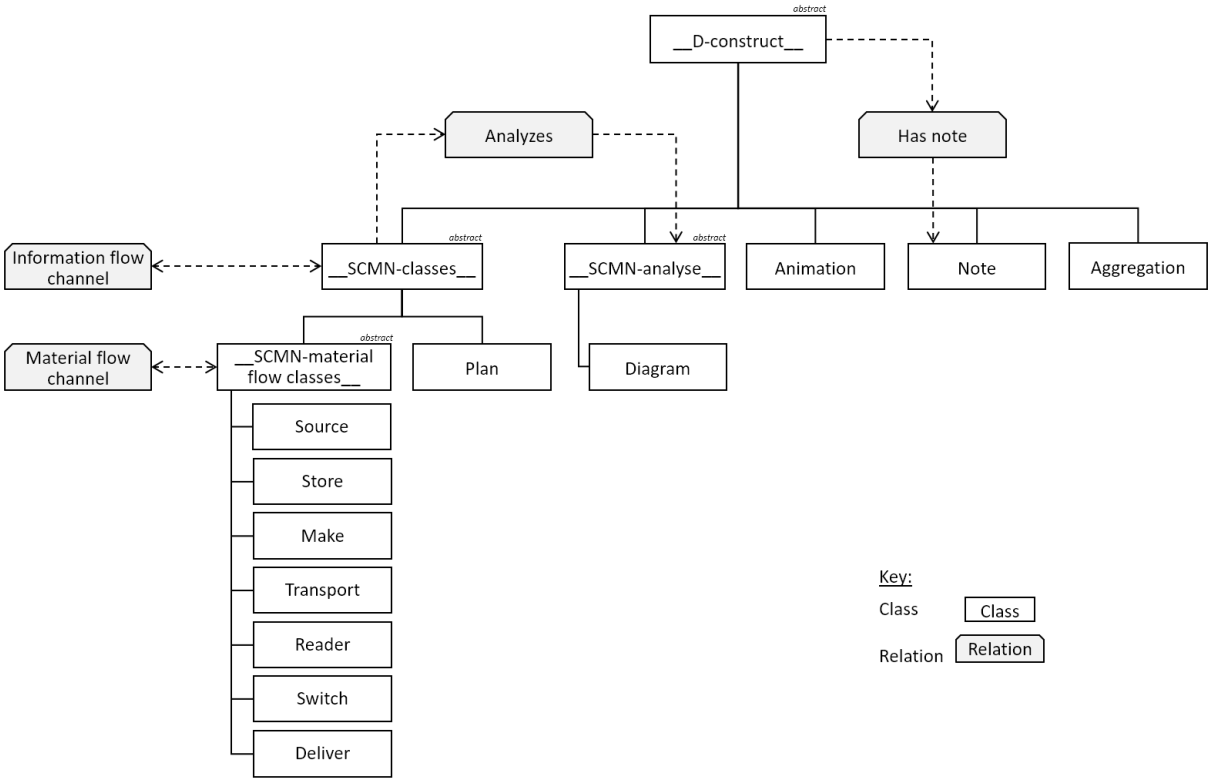


Abbildung 61: Klassendiagramm des Supply Chain Network Model

Die abstrakte Klasse _SCMN-classes_ wird von der Klasse __D-construct__ instanziiert. Alle Klassen die von _SCMN-classes_ abgeleitet werden, können mittels der Beziehung Information flow channel miteinander verbunden werden. Da die Beziehung Material flow channel nicht für die Modellierungsklasse Plan anwendbar ist, wurde die abstrakte Klasse __SCMN-material flow classes__ definiert, von der die statischen Objekttypen des Materialflusses abgeleitet werden. Relationen vom Typ Material flow channel können somit nur zwischen Objekten, die von der Klasse __SCMN-material flow classes__ instanziiert wurden, erstellt werden. Weitere Einschränkungen können mittels der Kardinalitätendefinition von Klassen gesetzt werden. Beispielsweise kann ein Source-Objekt keine eingehende Materialfluss-Relation haben. __SCNM-analyse__ und die davon abgeleitete Modellierungsklasse wird benötigt, um Ergebnisdiagramme im Modelleditor anzeigen zu können. Die Beziehung Analyzes kann benutzt werden, um einen Datenstrom von einem statischen Modellierungsobjekt zu einem Analyseobjekt zu erzeugen. Die Klasse Animation wird instanziiert, um den aktuellen Stand der Simulationsuhr und dynamische Objekte zu visualisieren. Die Klassen Note und Aggregation können vom Modellierer verwendet werden, um die Modelle mit verbalen Beschreibungen anzureichern, beziehungsweise Teile des Modells voneinander abzugrenzen.

Die Implementierung des Item Models folgt dem Klassenmodell dargestellt in Abbildung 62.

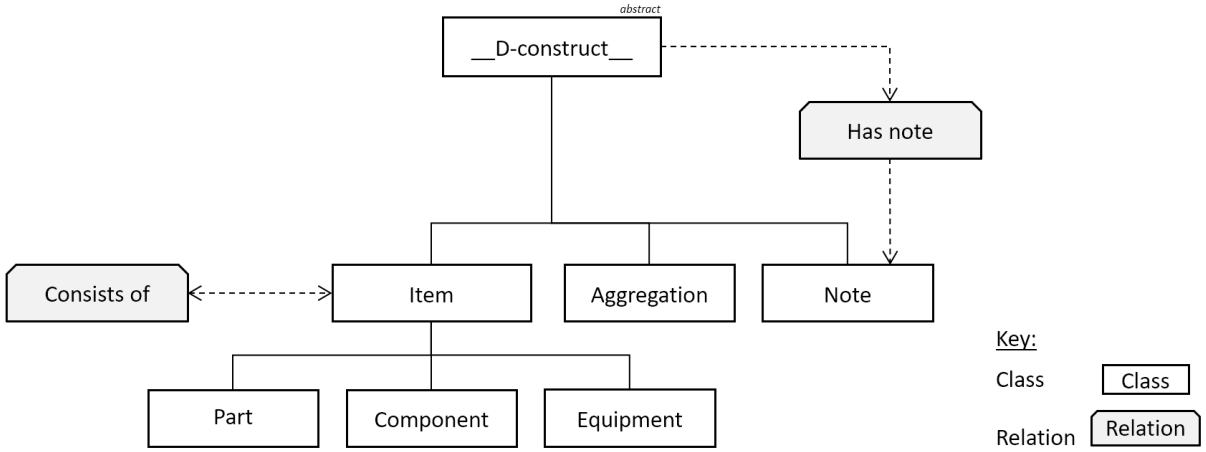


Abbildung 62: Klassendiagramm des Item Model

Sowohl Item, Part, Component und Equipment sind instanziierbare Klassen und können mittels der Relation Consists of miteinander in Beziehung gesetzt werden.

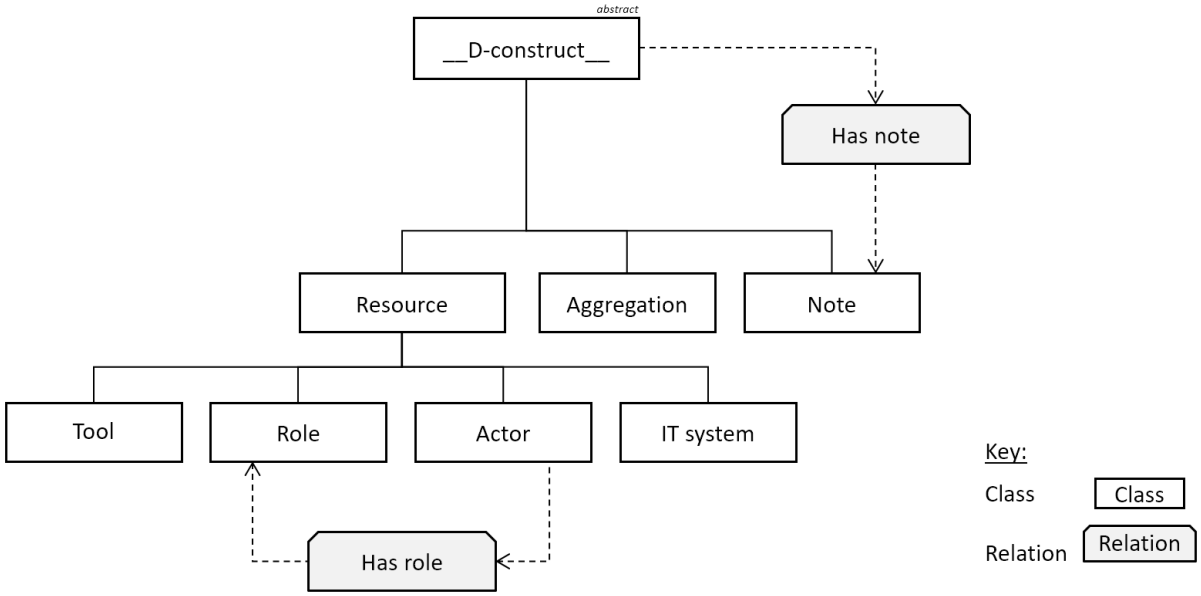


Abbildung 63: Klassendiagramm des Resource Model

Resource, Tool, Role, Akteur und IT-System sind im Modelleditor instanziierbar. Akteuren kann mittels der Relation Has role eine oder mehrere Rollen zugewiesen werden.

Im Information Model, siehe Abbildung 64, können die Klassen Information object, Message und Call / Request instanziiert werden.

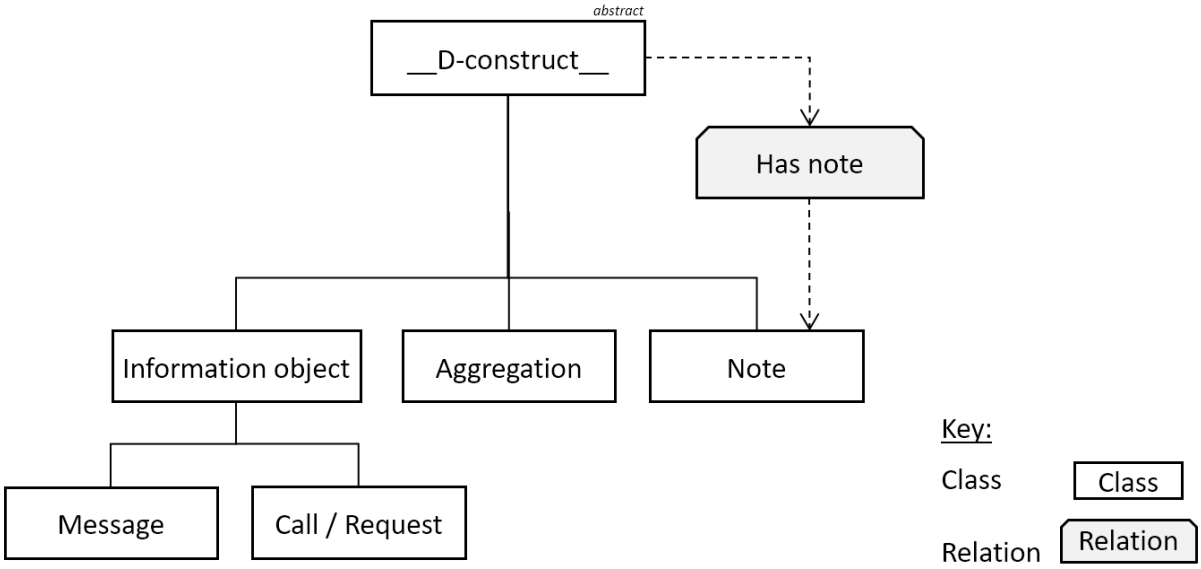


Abbildung 64: Klassendiagramm des Information Model

7.2.2 Definition der Modelltypen

Wie aus der Spezifikation des Metamodells in den vorhergehenden Kapitels ersichtlich, wurden die Modelltypen

- Supply Chain Network Model
- Resource Model
- Item Model
- Information Model

implementiert.

Im Folgenden wird nur die Implementierung des Supply Chain Network Modells beschrieben, da gemäß einigen, in diesem Kapitel beschriebenen Implementierungsentscheidungen, die restlichen Modelltypen zur Nutzung und Evaluierung der Methode nicht unbedingt erforderlich sind.

Der Modelltyp Supply Chain Network Model wird in ADOxx mittels der folgenden Konfiguration definiert.

```
001 MODELTYPE "Supply Chain Network Model" from:none plural:"Supply Chain
    Network Models" pos:1 not-simulateable
002 INCL "Plan"
003 INCL "Source"
004 INCL "Store"
005 INCL "Make"
006 INCL "Transport"
007 INCL "Reader"
008 INCL "Switch"
009 INCL "Deliver"
010 INCL "Diagram"
011 INCL "Animation"
012 INCL "Aggregation"
013 INCL "Note"
014 INCL "Material flow channel"
015 INCL "Information flow channel"
016 INCL "Analyzes"
017 INCL "Has note"
018
019 MODE "Standard" from:all
020 MODE "Dokumentation" from:all no-modeling
```

7.3 Implementierung der Attribute der Objekte

In ADOxx werden die Attribute einer Klasse zunächst in der sogenannten Klassenhierarchieverwaltung angelegt. Dabei wird der Name des Attributs, der Attributtyp, wenn erforderlich Standardwerte und Attributfacetten definiert. Um die Attribute im Modellierungseditor für den Modellierer zugänglich zu machen, muss zunächst ein sogenanntes Notebook definiert werden, das beim Doppelklick auf ein Modellierungsobjekt erscheint. Das Notebook und die darin enthaltenen Attribute werden im Klassenattribut AttrRep einer Klasse definiert. Der folgende Sourcecode zeigt dies am Beispiel der Klasse Make.

```
001 NOTEBOOK
002 CHAPTER "Description"
003 ATTR "Name"
004 ATTR "Order"
005 ATTR "Production program"
006 ATTR "Remaining production time"
007 ATTR "Production history"
008 ATTR "Description"
009 ATTR "External graphic"
010 ATTR "Id"
```

Das daraus resultierende Notebook ist in Abbildung 65 dargestellt.

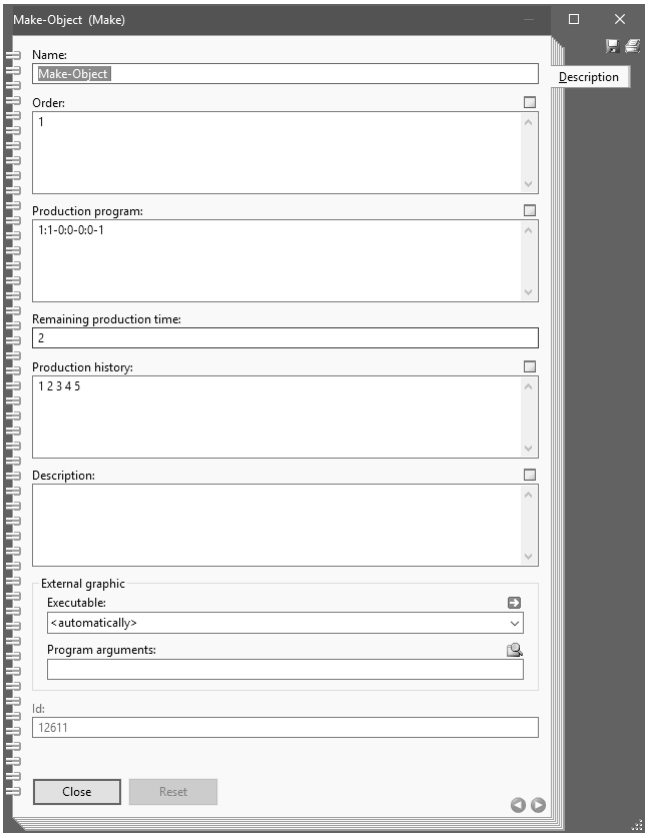


Abbildung 65: Notebook der Modellierungsklasse Make im ADOxx Modelleditor

7.4 Graphische Umsetzung der Notation

Nach dem in den bisherigen Kapitel das Metamodell der Methode und die Semantik der einzelnen Objekttypen herausgearbeitet wurden, wird in diesem Kapitel die visuelle Umsetzung der graphischen Modellierungssprache dargestellt. Moody (Moody, 2009) stellt in seinem Aufsatz fest, dass Entscheidungen bezüglich des Semantik der Sprachobjekte häufig mit großer Sorgfalt getroffen werden, aber die graphische Repräsentation dieser Objekte oftmals als trivial oder unwichtig betrachtet wird. Er zeigt im Weiteren, dass die Form der Darstellung zumindest einen gleich großen Einfluss auf die Effizienz des kognitiven Prozesses hat, wie der vermittelte Inhalt (Moody, 2009, S. 758). Ähnlich definiert Fill (Fill, 2009, S. 19) die Visualisierung von Modellelementen als "the use of graphical representations to amplify human cognition".

Da mit der Methode SIMchronization eine neue Modellierungssprache entwickelt wird, kann nicht auf einen bestehenden Visualisierungsstandard zurückgegriffen werden. Die graphische Notation muss deshalb neu entwickelt werden. Im Folgenden wird die graphische Umsetzung des Modelltyps Supply Chain Network Model besprochen.

7.4.1 Modellierungsrichtung

Bertin (Bertin, 1983) definiert acht visuelle Variablen mit denen Information mit graphischen Mitteln dargestellt werden kann. Bei flussorientierten Modellierungssprachen, die auf einer zwei-dimensionalen Zeichenfläche angewendet werden, zeigt die horizontale und vertikale Position der Modellierungsobjekte untereinander üblicherweise eine Reihenfolge der Abarbeitung dar. Dies ist von besonderer Bedeutung, wenn diese Reihenfolge nicht durch ein Beziehungsobjekt, wie beispielsweise einer „Nachfolge"-Relation in ADONIS BPMS definiert wird, sondern durch die Positionierung der Objekte auf der Modellierungsfläche, wie das bei mittels Chevrons modellieren Value-Chains der Fall ist.

Die Frage, ob die Modelle einer Modellierungsmethode besser horizontal oder vertikal dargestellt werden, kann nicht eindeutig beantwortet werden. Der Erfahrung des Autors nach werden Modelle, die im Intranet publiziert werden und auf die über einen Web-browser zugegriffen wird, besser von oben nach unten modelliert, da dies dem typischen Scroll-Verhalten der User für Webseiten entspricht. Prozesse, die in Qualitätshandbücher übernommen werden sind ebenfalls einfacher im Hochformat zu verarbeiten. Werden die Modelle aber für einen Workshop auf DIN A0 ausgedruckt und an die Wand gepinnt, so hat sich eher die waagrechte Modellierung bewährt. Außerdem wird bei einer horizontalen Modellierung die am Monitor zur Verfügung stehende Fläche besser genutzt, was vor allem während der Animation ein Vorteil ist[74]. EPK-Modelle werden üblicherweise von oben nach unten modelliert, wohingegen BPMN-Modelle häufiger von links nach rechts dargestellt werden. ADONIS BPMN wird sowohl horizontal als auch vertikal angewendet. Die Software unterstützt sogar eine automatische Transformation der Modellierungsrichtung der Modelle.

Da der Hauptanwendungszweck der Methode SIMchronization die Transparentmachung von dynamischen Zusammenhängen in einer Supply Chain ist und ein wichtiges Mittel dazu die Simulation und Animation ist, wird die Methode für eine horizontale Modellierungsrichtung optimiert. Sie kann jedoch auch senkrecht angewendet werden.

7.4.2 Graphische Darstellung der Modellierungsobjekte

Neben der beiden planaren Variablen, definiert Bertin (Bertin, 1983) noch sechs weitere graphische Konzepte zur Konstruktion graphischer Notationen, die Form, Größe, Farbe, Helligkeit, Ausrichtung, Textur. Die für SIMchronization entwickelten graphischen Objekte werden sich nur in Farbe und Form unterscheiden. Die Größe einzelner Objekte ist in der Methode vordefiniert, kann aber vom Modellierer verändert werden. Gleichartige

[74] Offensichtlich gilt dieses Argument nur für Monitore und Grafiksysteme die ausschließlich im Querformat betrieben werden können

Objekttypen sind standardmäßig gleich groß. Der Farbton wird aus der SCOR Implementierung in ADOlog® übernommen, allerdings viel heller gewählt, um die Lesbarkeit eines schwarz-weiß Ausdrucks zu gewährleisten. Die Ausrichtung ist fest von der Methode definiert und kann vom Modellierer nicht verändert werden. Die Textur der Füllung der einzelnen Objekttypen ist unveränderlich vollflächig.

7.4.2.1 Implementierung der Graphischen Notation am Beispiel des Make-Objekts

Objekte des Typs Make sind der zentrale Baustein einer Maintenance Supply Chain. Teile werden von Make-Objekten verändert, verbraucht oder erzeugt und damit Wertschöpfung erzeugt. Werden mehrere dieser Objekte miteinander verknüpft, entsteht eine Wertschöpfungskette. Graphisch wird dieser Bausteingedanke durch die Anlehnung des Designs an LEGO®-Bausteine umgesetzt, die miteinander verbunden werden können. In Abbildung 66 ist links ein Objekt zu sehen, wie es direkt nach der Instanziierung auf der Modellierungsfläche erscheint und rechts ein Make-Objekt während der Simulation. Dabei ist das Attribut „Production program“ mit dem Wert „1:1-0:0-0:0-1“ belegt und das Attribut „Remaining production time“ mit dem Wert 2.

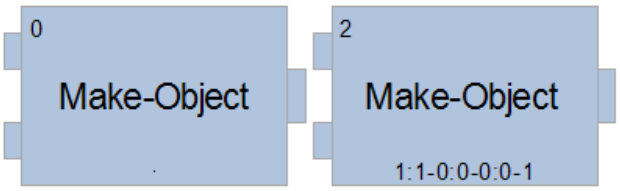


Abbildung 66: Graphische Umsetzung des Make-Objekts

In ADOxx wird die graphische Darstellung mittels der LEO-Sprache im Klassenattribut „Graphrep“ jeder Klasse erzeugt.

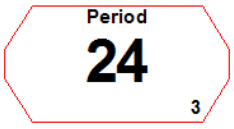
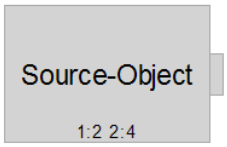
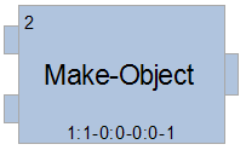
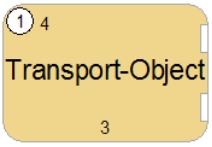
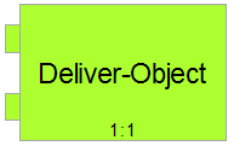
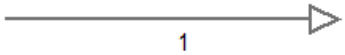
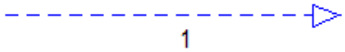
```
001 GRAPHREP sizing:asymmetrical smart-symbol-size
002 SHADOW off
003 PEN w:0.01cm color:darkgray
004 FILL color:lightsteelblue
005
006 TABLE x:-1.5cm y:-1cm w:3cm h:2cm cols:3 rows:3 w1:0.3cm w2:100%
    w3:0.3cm h1:0.3cm h2:100% h3:0.3cm
007 STRETCH off
008 RECTANGLE x:(tabx0) y:(taby0) w:(tabw1+tabw2+tabw3)
    h:(tabh1+tabh2+tabh3)
009 RECTANGLE x:(tabx0-0.2cm) y:(taby1) w:(0.2cm) h:(0.4cm)
010 RECTANGLE x:(tabx0-0.2cm) y:(taby2-0.4cm) w:(0.2cm) h:(0.4cm)
011 RECTANGLE x:(tabx3) y:((taby3+taby0)/2-0.3cm) w:(0.2cm) h:(0.6cm)
012
013 FONT h:12pt
014 ATTR "Name"  w:c:3cm h:c:1.5cm line-break:words line-height:12pt
015
016 FONT h:8pt
017 ATTR "Production program" x:(0) y:(taby3 - 0.3cm) w:c:3cm h:t:0.3cm
    line-break:words line-height:8pt
018 ATTR "Remaining production time"  x:(tabx0 + 0.1cm) y:(taby0 + 0.1cm)
    w:l h:t
```

Dem Design der restlichen Objekttypen liegen folgende Überlegungen zu Grunde. Das Plan-Objekt ist wie ein Ereignis in der EPK-Modellierungssprache gestaltet. Es zeigt, dass dieses Objekt Zustandsänderungen selbsttätig auslösen und auf Ereignisse reagieren kann. Das Store-Objekt ähnelt in abstrakter Weise einer Palette, das Transport-Objekt einem Lastwagen. Das Source-Objekt einer Batterie, die den Fluss speist und Deliver einem Prellbock, der das Ende der Supply Chain markiert. Das Reader-Objekt hat die Form eines Gate-Readers (Heiserich et al., 2011, S. 351) und der Switch ähnelt einem Prisma das einen Strahl in verschiedene Richtungen lenkt. Material flow channel sind als durchgängige Linie ausgeführt und die gestrichelte Linie des Information flow channel zeigt, dass kein physisches Gut auf diesem Kanal transportiert wird.

7.4.2.2 Graphische Umsetzung der Modellierungsobjekte des Supply Chain Network Models

Tabelle 15 zeigt die Visualisierung der zentralen Objekttypen des Supply Chain Network Modells. Die Klassen Note und Aggregation wurden in dieser Liste nicht aufgeführt, da deren Implementierung aus der Modellierungssprache ADONIS BPMS übernommen wurde.

Tabelle 15: Graphische Repräsentation der Modellierungsobjekte des SCNM

Graphische Repräsentation	Visualisierte Attribute mit Beispielwerten
	Timestamp (24) Micro step (3)
	Name (Source-Object) Source program (1:2 2:4)
	Item type (1) Name (Store-Object) Numbers of items in stock (3)
	Remaining production time (2) Name (Make-Object) Production program (1:1-0:0-0:0-1)
	Item type (1) Numbers of items arrived (4) Name (Transport-Object) Transport time (3)
	Name (Reader-Object)
Switch-Objekt 	
	Name (Deliver-Object) Deliver program (1:1)
Material flow channel 	Index (1)
Information flow channel 	Index (1)

7.4.2.3 Visualisierung während der Simulation und Animation

Rohrer (Rohrer, 2000) diskutiert in seiner Arbeit die Elemente einer guten graphischen Umsetzung von Animation von Produktionsabläufen. Er listet dabei Interaktivität, Realismus, Performanz, Flexibilität und Einfachheit der Benutzung auf. Die Implementierung wird der Anforderung nach Interaktivität gerecht, als dass sie eine schrittweise Simulationsfunktionalität bietet, die es dem Betrachter ermöglicht, nach jedem simuliertem Zyklus auf Modellinhalte zugreifen zu können. Im Gegensatz zu kommerziellen Tools, die Produktionsverfahren in Fabriken simulieren und die verwendeten Anlagen und Transportmodule realistisch nachbilden, bleibt die Methode SIMchronization bewusst abstrakt. Dennoch können statische und dynamische Objekte bei Bedarf durch Bilder ersetzt werden. Performanz ist für die, in dieser Arbeit beschriebene prototypische Umsetzung von nachrangiger Priorität. Flexibilität der Darstellung der Animation ist durch die Metamodellierungsfunktionalität der Plattform prinzipiell gegeben und Einfachheit der Benutzung wird durch die Verwendung einer einheitlichen Benutzerschnittstelle unterstützt.

Abbildung 67 zeigt links ein Store Objekt während der Simulation. Die live-Animation zeigt dabei über dem Objekt die IDs der im Store-Objekt aktuell gelagerten Items an und unter dem Objekt die IDs, der sich im Einlagerungsprozess befindlichen Items. Die mittlere Darstellung zeigt das Store-Objekt während der offline-Animation. Rechts in der Abbildung ist ein Item zu sehen, welches die aktuelle Microstep Nummer, seinen Itemtyp und seine Item ID anzeigt. Items verschiedenen Typs werden in unterschiedlichen Farben dargestellt.

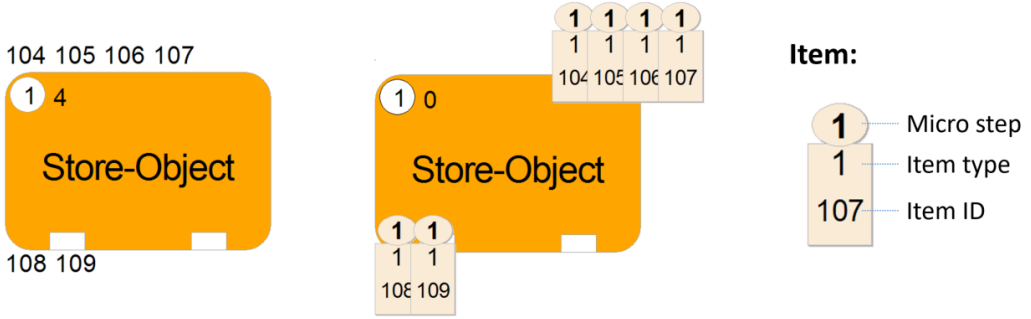


Abbildung 67: Das Store-Objekt während der Simulation und Animation und ein Item

Neben der, in Abbildung 67 dargestellten, dynamischen Objekte des Materialflusses werden die dynamischen Objekte des Informationsflusses an den Beziehungen des Typs Information flow channel dargestellt. Ein Beispiel für die Darstellung eines Get / Request und Message Informationsobjekts direkt am Information flow channel zeigt Abbildung 68.

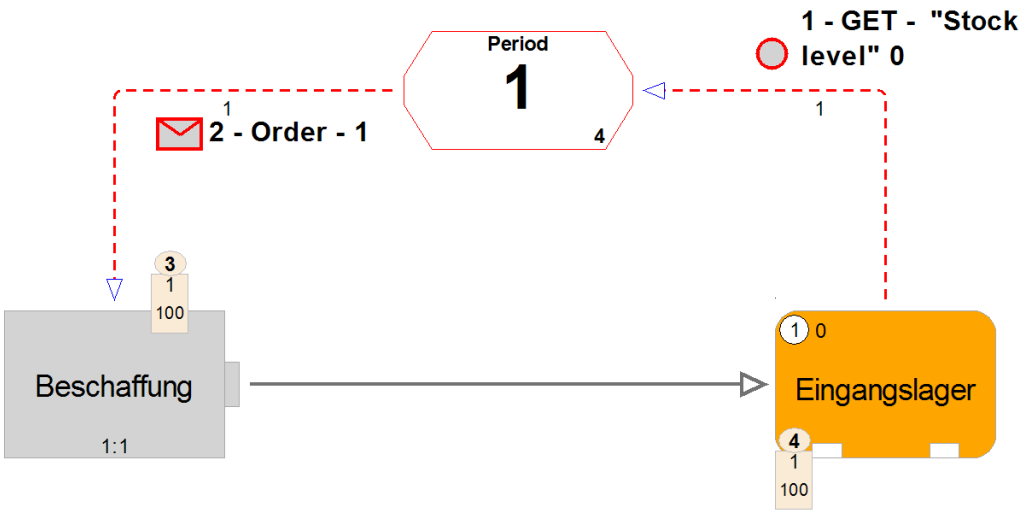


Abbildung 68: Informationsobjekte versendet über Information flow channels

Im ersten Microstep des Beispiels erhält das Plan-Objekt den aktuellen Lagerstand (im Beispiel 0) des Eingangslagers als Antwort auf den Get / Request an das Store-Objekt. Das Plan-Objekt sendet daraufhin im zweiten Microstep eine Bestellung per Nachricht (im Beispiel ist die Bestellmenge 1) an das Source-Objekt, welches ein Item mit der ID 100 beschafft und diesen unverzüglich im vierten Microstep zur Einlagerung übergibt.

7.5 Implementierung des Regel-Engine

Ein Grund für die Entscheidung, die Methode SIMchronization vollständig in ADOxx umzusetzen ist, dass die Plattform eine Sprache zur Definition von sogenannten Ausdrücken[75] zur Verfügung stellt. Ausdrücke sind sehr ähnlich zu dem, aus Tabellenkalkulationsprogrammen bekannten Konzept der zellenbasierten Formel, die eine laufend aktualisierte Berechnung durchführt und dabei Werte aus anderen Zellen nutzen kann. In ADOxx können diese Werte beispielsweise aus Attributen von Objekten oder Modellen herangezogen werden.

Die Metamodellierungsplattform ADOxx verfügt mit LEO über eine Meta-Sprache, deren Ausprägungen an verschiedenen Stellen in ADOxx verwendet werden, wie beispielsweise in Notebook- oder Modelltypdefinitionen. Teil von Leo sind die LeoExpressions, eine Sammlung basaler Funktionen und Operatoren (BOC Information Technologies Consulting AG, 2014), wie zum Beispiel:

- Logische Operatoren
- Vergleichsoperatoren
- Arithmetische Operatoren und Funktionen

[75] In Englisch „expressions“

- Zeichenketten Operatoren und Funktionen
- Operatoren und Funktionen für verschiedene Datenstrukturen und Datentypkonversionen

Eine Erweiterung der LeoExpressions stellen die CoreExpressions dar. Sie können auf den Core von ADOxx und somit auf das Modellrepository zugreifen und beispielsweise Werte von Objektattributen auslesen oder ändern. Eine dritte Art der Ausdrücke sind die AdoScriptExpressions, welche in ADOscript vom Programmierer definiert werden und in der Anwendung anschließend verwendet werden können.

Diese sehr mächtige und flexible Art und Weise Formeln zu erstellen, kann auch zur Definition von Regeln verwendet werden. Da die Ausdrücke auf das Repository zugreifen können, ist keine separate Datenhaltung der Simulationsergebnisse notwendig. Die Simulationsergebnisse können vom Simulationsalgorithmus direkt in ein Objekt oder Modell gespeichert werden und die notwendigen Datenbanktransaktionen werden von ADOxx übernommen. Die Fact Base für die Regeln ist somit sowohl für strukturelle, als auch historische Daten das simulierte Modell.

Um eine einfache Production Rule in ADOxx zu erstellen wird der Befehl *cond* verwendet.

Syntax – Grammatik:

```
cond ( Bedingung 1, Ausdruck 1,
        … ,
        Bedingung n, Ausdruck n,
        AlternativerAusdruck n + 1
      )
```

Der gesamt Umfang der Expression-Sprache von ADOxx ist in (BOC Information Technologies Consulting AG, 2014) beschrieben. Um die Regelsprache an die besonderen Erfordernisse von SIMchronization anzupassen, wurden unter anderem die folgenden Erweiterungen vorgenommen:

Variablen:

Tabelle 16: Ausgewählte zusätzliche Variablen für SIMchronization

Variable	Beschreibung
TIMESTAMP	enthält die aktuelle Simulationszeit
ITEM	Item ID eine dynamischen Objekts des Materialflusses
INFOIN1, INFOIN2, INFOIN3	verwendet in einer Regel eines Objekts liefert es die ID des adjazenten Objekts eines eingehenden Information flow channel mit Index = 1, 2 oder 3
INFOOUT1, INFOOUT2, INFOOUT3	verwendet in einer Regel eines Objekts liefert die Objekt-ID des adjazenten Objekts eines ausgehenden Information flow channel mit Index = 1, 2 oder 3

Funktionen:

Tabelle 17: Ausgewählte zusätzliche Funktionen für SIMchronization

Funktion	Beschreibung
get ()	Wird zur Umsetzung eines Call / Requests, also einer synchronen Anfrage eines Objekts an ein anderes Objekt verwendet, dabei wird folgende Syntax für die Regelsprache definiert: *get (Objekt-Identifizierer, Attributname)* Der Objekt-Identifizierer ist dabei die Objekt-ID des abzufragenden Objekts, diese kann beispielsweise über die Verknüpfung mit einem Information flow channel und Abfrage mittels INFOIN1 ermittelt werden.
send()	Als Array implementiert, werden Inhalte, die einer send-Variable zugewiesen werden als Nachricht versendet. Eine Nachricht besteht aus folgenden Komponenten, die in einem String mit Trennzeichen übergeben werden: • Empfänger Objekt ID • Message Identifier: zur Unterscheidung verschiedener Nachrichtentypen • Nachrichteninhalt
READ	Liest einen Speicherplatz eines Items aus. Wird unter anderem zur Umsetzung der Auto-ID Funktionalität verwendet.
WRITE	Schreibt einen Wert in einen Speicherplatz eines Items

7.5.1 Syntax eines Produktionsprogramms

Ein Produktionsprogramm wird in das Attribut “Production program” eines Make-Objekts eingetragen und legt die Tätigkeitsart, den Ressourcenverbrauch und das Ergebnis eines Make-Prozesses fest.

7.5.1.1 Produktionsprogramm für Produktion oder Bearbeitung

Das Produktionsprogramm gibt an, welche Kombination von Inputs nach einer definierten Bearbeitungszeit zu welcher Kombination von Outputs führt. Inputs sind dabei bearbeitete und verbrauchte Teile. Outputs sind bearbeitete und erzeugte Teile.

Naheliegend wäre, die Kombination aus bearbeiteten, erzeugten und verbrauchten Teilen als Tabelle in ADOxx abzubilden. Wie im Beispiel in Tabelle 18 ersichtlich.

Tabelle 18: Variante für die Angabe eines Produktionsprogramms

Bearbeitete Teile		Verbrauchte Teile		Erzeugte Teile		Bearbeitungszeit
Anzahl	Typ	Anzahl	Typ	Anzahl	Typ	
0	0	2	1	1	3	2
		1	2			

Diese Tabelle könnte mittels den in ADOxx zur Verfügung stehenden Attributtyp Record umgesetzt werden. Die Anzeige im Objektsymbol Make gestaltet sich aber schwierig. Außerdem ist die Veränderung eines Programms das über mehrere Tabellenzellen hinweg beschrieben wird, relativ aufwendig. Aus diesen Gründen wurde für das Produktionsprogramm eine eigene Syntax entwickelt, welche die Input-/Output-Faktoren und die Bearbeitungszeit in Tupel-Form abbildet.

Die Syntax eines Produktionsprogramms als Erweiterte Backus-Naur-Form:

```
Teiletyp = ZifferAußerNull { Ziffer } | "0" ;
Anzahl = ZifferAußerNull { Ziffer } | "0" ;
TeiletypAnzahl = Teiletyp":"Anzahl ;
BeabeiteteTeile = TeiletypAnzahl{" "TeiletypAnzahl} ;
VerbrauchteTeile = TeiletypAnzahl{" "TeiletypAnzahl} ;
ErzeugteTeile = TeiletypAnzahl{" "TeiletypAnzahl} ;
Bearbeitungszeit = ZifferAußerNull { Ziffer } ;
Produktionsprogramm = BeabeiteteTeile"-"VerbrauchteTeile"-
"ErzeugteTeile"-"Bearbeitungszeit ;
```

Diese Syntax erlaubt die Angabe des Produktionsprogramms in kompakter Form. Neben den Teilen ist auch die, für die Durchführung dieses Produktionsschritts notwendige Bearbeitungszeit enthalten.

7.5.1.2 Produktionsprogramm für Verpackungsvorgänge

Ein Make-Objekt kann dazu verwendet werden, ein oder mehrere Teile in ein anderes Teil zu verpacken, diese Anweisung wird über ein spezielles Produktionsprogramm gegeben.

Ein Beispiel hierfür ist: *PACK-2:1-1:6-1*

Die Syntax eines Produktionsprogramms für Verpackung als Erweiterte Backus-Naur-Form:

```
Teiletyp = ZifferAußerNull { Ziffer } ;
Anzahl = ZifferAußerNull { Ziffer }   | "0" ;
TeiletypAnzahl = Teiletyp":"Anzahl ;
```

```
VerpackungsTeil = Teiletyp":1" ;

ZuVerpackendeTeile = TeiletypAnzahl{" "TeiletypAnzahl} ;

Bearbeitungszeit = ZifferAußerNull { Ziffer } ;

ProduktionsprogrammVerpackung = "PACK-"VerpackungsTeil"-"
ZuVerpackendeTeile"-"Bearbeitungszeit ;
```

7.5.1.3 Produktionsprogramm für Vorgänge des Entpackens

Zum Entpacken eines Behälters wird ebenfalls ein spezielles Produktionsprogramm angewendet.

Sollen in einem Schritt alle verpackten Teile entpackt werden, so wird folgendes Produktionsprogramm angewendet:

Beispiel: *UNPACK_all-2:1-1*

Die Syntax als Erweiterte Backus-Naur-Form:

```
Teiletyp = ZifferAußerNull { Ziffer } ;

Anzahl = ZifferAußerNull { Ziffer }  | "0" ;

TeiletypAnzahl = Teiletyp":"Anzahl ;

VerpackungsTeile = TeiletypAnzahl{" "TeiletypAnzahl} ;

Bearbeitungszeit = ZifferAußerNull { Ziffer } ;

ProduktionsprogrammVerpackung = "UNPACK_all-"VerpackungsTeile"-"
Bearbeitungszeit ;
```

Dieses Programm hat den Vorteil, dass die Anzahl der verpackten Teile eines Behälters nicht vor der Auftragserteilung an ein Make-Objekt bekannt sein muss.

Sollen nur einzelne Teile aus einem Behälter entnommen werden, so ist folgendes Produktionsprogramm anzuwenden.

Beispiel: *UNPACK-2:1-1:3-1*

Die Syntax als Erweiterte Backus-Naur-Form:

```
Teiletyp = ZifferAußerNull { Ziffer }  ;

Anzahl = ZifferAußerNull { Ziffer }  | "0" ;

TeiletypAnzahl = Teiletyp":"Anzahl ;

VerpackungsTeile = TeiletypAnzahl ;

VerpackteTeile = TeiletypAnzahl{" "TeiletypAnzahl} ;

Bearbeitungszeit = ZifferAußerNull { Ziffer } ;

ProduktionsprogrammVerpackung = "UNPACK-"VerpackungsTeil"-
"VerpackteTeile "-"Bearbeitungszeit ;
```

Das Attribut „Production program“ kann mit Regeln hinterlegt werden. Somit können sich Produktionsprogramme im Zeitverlauf oder je nach Steuerungsinformation verändern. Eine einfache Regel wäre, dass nach Ablauf einer gewissen Zeitspanne durch den Lerneffekt eine beschleunigte Bearbeitung erfolgt:

Regel:

```
cond (TIMESTAMP = 20, „0:0-1:1 2:3-3:1-5“, „0:0-1:1 2:3-3:1-4“)
```

Ist die verbleibende Bearbeitungszeit größer 1, so müssen dennoch alle Input-Teile in der ersten Periode zur Verfügung stehen und die produzierten Teile werden erst in der letzten Periode in das nachfolgende statische Supply Chain Network Objekt geschoben.

7.5.2 Syntax eines Einkaufsprogramms

Wie beim Make-Prozess erfolgt die Definition der zu liefernden Teiletypen und Mengen durch ein Programm, dem sogenannten Einkaufsprogramm.

Die Syntax eines Einkaufsprogramms als Erweiterte Backus-Naur-Form:

```
Teiletyp = ZifferAußerNull { Ziffer } ;
Anzahl = ZifferAußerNull { Ziffer }  | "0" ;
TeiletypAnzahl = Teiletyp":"Anzahl ;
ZuBeschaffendeTeile = TeiletypAnzahl{" "TeiletypAnzahl} ;
Einkaufsprogramm = ZuBeschaffendeTeile ;
```

7.5.3 Definition eines Lieferprogramms

Entsprechend dem Make- und Source-Prozess erfolgt die Definition der an den Kunden zu liefernden Teiletypen und Mengen durch ein Programm.

Die Syntax eines Lieferprogramms als Erweiterte Backus-Naur-Form:

```
Teiletyp = ZifferAußerNull { Ziffer } ;
Anzahl = ZifferAußerNull { Ziffer }  | "0" ;
TeiletypAnzahl = Teiletyp":"Anzahl ;
ZuLieferndeTeile = TeiletypAnzahl{" "TeiletypAnzahl} ;
Lieferprogramm = ZuLieferndeTeile ;
```

7.6 Implementierung des Simulationsalgorithmus

Wie im Theorieteil zur Simulation erläutert, kann ein Simulationsalgorithmus periodenorientiert oder ereignisgesteuert sein. Vor- und Nachteile beider Methoden sind in Kapitel 2.5.5.1 und 2.5.5.2 aufgeführt. Wenngleich für eine kommerzielle Umsetzung der Methode SIMchronization eine ereignisgesteuerte Simulation verwendet werden sollte, wurde vom Autor entschieden, für die prototypische Umsetzung auf ADOxx einen periodenorientierte Simulationsalgorithmus zu implementieren. Das Hauptargument dafür ist die einfachere und schnellere Implementierung sowohl des Algorithmus selbst, als auch der Umsetzung der verhaltensdefinierenden Regeln und ihrer Faktenbasis. Die Evaluation der Methode kann auf Basis beider Implementierungsalternativen erfolgen. Der Autor hat mit der Umsetzung des hybriden Simulationsalgorithmus, beschrieben in Prackwieser et al. (Prackwieser et al., 2013) gezeigt, dass auf der Plattform ADOxx auch die Implementierung eines ereignisgesteuerten diskreten Simulationsalgorithmus möglich ist.

Die Simulation kann auf zweierlei Weise durchgeführt werden. Wie in Abbildung 69 ersichtlich, kann die Simulation schrittweise oder kontinuierlich bis zum Ende ausgeführt werden. Die schrittweise Vorgehensweise hat den Vorteil, dass der Benutzer die Simulation nach Ende jeder Periode abbrechen kann, auf Zwischenergebnisse zugreifen und bei Bedarf diese manuell verändern kann. Nach der Simulation können unterschiedliche Analyseverfahren durchlaufen werden. Aktuell ist ein Modul zum zeitbasierten durchsteppen des Supply Chain Netzwerks vorhanden und ein weiteres, das den Lebenszyklus eines dynamisches Objekt des Materialflusses aufzeigt.

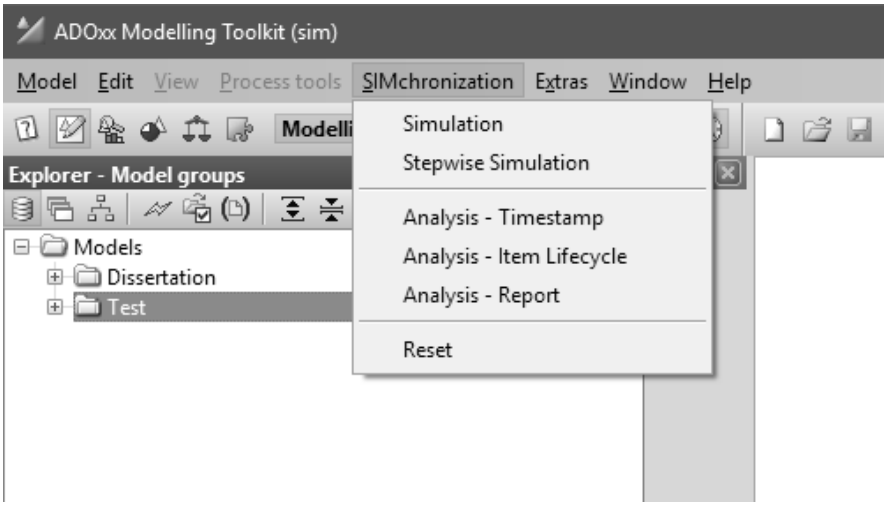


Abbildung 69: Startmenu

In der nachfolgenden Beschreibung werden die wichtigsten Konstrukte des Sourcecodes aufgeführt. Die Methode wird auf der OpenSource Plattform OMiLAB zur Verfügung gestellt. Da der Algorithmus in ADOscript implementiert wurde kann der Sourcecode dort eingesehen werden.

Nach Auswahl der Simulation im Startmenü wird der Benutzer nach der Anzahl der zu simulierenden Perioden gefragt. Es wird immer das aktuell geöffnete Supply Chain Network Model simuliert. Die vom Nutzer angegebene Periodenanzahl wird in jedem Fall durchlaufen, unabhängig ob dynamische Objekte existieren oder nicht.

Zunächst erfolgt die Initialisierung des Simulationsmodells. Während der Initialisierung werden die benötigten Datenstrukturen aufgebaut. Unter anderem wird die Struktur des Supply Chain Network Modells, also dessen statische Objekte und deren Material- und Informationsflussbeziehungen in ein Datenmodell übertragen. Dies geschieht zum einen dazu, die Performanz des Simulationsalgorithmus zu verbessern, da dadurch weniger Zugriffe auf den ADOxx-Core und das Modellrepository notwendig sind. Zum anderen kann diese Datenstruktur, wie im konzeptionellen Modell vorgesehen, die Basis für eine Schnittstelle zu einem spezialisierten Simulationstool sein. In der vorliegenden prototypischen Implementierung werden nicht alle zur Simulation notwendigen Daten in die Datenstruktur übertragen, so werden auf viele Attributinhalte direkt über die Core-Interfaces zugegriffen. In einer weitergehenden Implementierung sollten auch diese Inhalte in die Datenstruktur überführt werden.

Der Simulationsalgorithmus wird getrieben durch die Simulationsuhr, deren Zähler Periode für Periode um 1 vorgerückt wird. In jeder Periode werden alle statischen Objekte der Typen Plan, Source, Deliver und Make durchlaufen und die dynamischen Objekte des Materialflusses entsprechend deren Verhalten bewegt. Wird ein dynamisches Objekt nach einem Source oder Make-Objekt durch einen Reader, Switch und/oder in einen Store oder Transport geschoben, so erfolgt diese Verarbeitung unverzüglich. Versendete Nachrichten an Source- und Make-Objekte werden in einem Messageboard zwischengespeichert und beim nächsten Durchlauf des Algorithmus abgearbeitet. Nach jedem Nachrichtenversand wird das Regelset des reaktiven Teils aller Plan-Objekte ausgewertet, um eine unverzügliche Steuerungsreaktion zu ermöglichen. Alle Get / Request-Anfragen werden vom Algorithmus unverzüglich beantwortet. Die folgenden Pseudo-Code Snippets beschreiben zentrale Teilaspekte des Simulationsalgorithmus.

Im ersten Code Block wird der Simulationsalgorithmus erläutert, nachfolgend dazu einige zentrale Prozeduren.

```
001 iNumberOfPeriods:( USER_ENTRY )
002 WHILE (iCurrentSimulationPeriod <= iNumberOfPeriods)
003 {
004   UPDATE_STORES
005   FOR_ALL PlanObjects
006   {
007     sRuleResult:( EVALUATE_RULE (
                        GET_OBJECT_ATTRIBUTE_VALUE ( "Action" ))
008     IF (sRuleResult contains Message)
009     {
010        Message.send;
011     }
012   }
013     SIMULATE_SOURCE
014   IF ('Material flow channel' next object type =
                                  "Reader" OR "Switch") THEN
015   {
016     SIMULATE_READER_SWITCH
017     MESSAGE_CHECK_PLAN_REACTION
018   }
001     SIMULATE_DELIVER
002     SIMULATE_MAKE
003   IF ('Material flow channel' next object type =
                                  "Reader" OR "Switch") THEN
004   {
005     SIMULATE_READER_SWITCH
006     MESSAGE_CHECK_PLAN_REACTION
007   }
008   iCurrentSimulationPeriod:( iCurrentSimulationPeriod + 1 )
009 }
010
```

Die verwendeten Prozeduren werden nachfolgend beschrieben:

```
001 PROCEDURE UPDATE_STORES
002 FOR_ALL StoreObjects
003 ( move items received last period into store )
004 FOR_ALL TransportObjects
005 ( update arrived item list )
```

```
001 PROCEDURE MESSAGE_CHECK_PLAN_REACTION
002 FOR_ALL PlanObjects
003 {
001   Check for incoming messages
002   IF (Message.received = TRUE) THEN
003   {
004     sMessageContent:(Message.content)
005     sRuleResult:( EVALUATE_RULE (
                        GET_OBJECT_ATTRIBUTE_VALUE ( "Reaction" ))
006     IF (sRuleResult contains Message)
007     {
008        Message.send;
009     }
010   }
011 }
```

```
001 PROCEDURE SIMULATE_SOURCE
002 FOR_ALL SourceObjects
003 {
004   Check for incoming messages
005   IF (Message.received = TRUE) THEN
006   {
007     iOrderVolume:(Message.content)
008   }
009   ELSE
010   {
011     iOrderVolume:( EVALUATE_RULE (
                        GET_OBJECT_ATTRIBUTE_VALUE ( "Order" )))
012   }
013   IF ( iOrderVolume > 0 ) THEN
014   {
015     sSourcingProgram:( GET_OBJETCT_ATTRIBUTE_VALUE
                             ( "Sourcing program" ))
016     FOR sSourcingPerItemType IN:( sSourcingProgram )
017     {
018       iNumberOfSourcedItems:( VAL token( sSourcingPerItemType, 1, ":" ))
019       iNumberOfItemsToCreate:( iNumberOfSourcedItems * iOrderVolume )
020       FOR iNumber FROM:( 1 ) TO:( iNumberOfItemsToCreate )
021       {
022         Item.create;
023         Item.settype( VAL token( sSourcingPerItemType, 0, ":" ))
024         Item.move( follow 'Material flow channel' to next object )
025       }
026     }
027   }
028 }
```

```
001 PROCEDURE SIMULATE_DELIVER
002 FOR_ALL DeliverObjects
003 {
004   tPrecedingStores:( GET_STORE_OBJECTS_PRECEDING_DELIVER )
005   iOrderVolume:( EVALUATE_RULE (
                        GET_OBJECT_ATTRIBUTE_VALUE ( "Order" )))
006   IF ( iOrderVolume > 0 ) THEN
007   {
008     iReadyForDeliver:(1)
009     sDeliveryProgram:( GET_OBJETCT_ATTRIBUTE_VALUE
                             ( "Delivery program" ))
010     FOR sDeliverPerItemType IN:( sDeliveryProgram )
011     {
012       iNumberOfItemsToDeliver:( iOrderVolume *
                                  VAL token(sDeliverPerItemType, 1, ":" ))
013       FOR iNumber FROM:(1) TO:( iNumberOfItemsToDeliver )
014       {
015         IF ( ALL_ITEMS_ARE_AVAILABLE_IN_STORES = FALSE )
016         {
017            iReadyForDeliver:(0)
               EXIT_THIS_PROCEDURE
018         }
          }
019     }
020     IF ( iReadyForDeliver = 1 ) THEN
021     {
022       FOR sDeliverPerItemType IN:( sDeliveryProgram )
023       {
024         iNumberOfItemsToDeliver:( iOrderVolume *
                                  VAL token(sDeliverPerItemType, 1, ":" ))
025         iTypeOfItemToDeliver:( VAL token(sDeliverPerItemType, 0, ":" ))
026
027         FOR oStoreObject IN:( tPrecedingStores )
028         {
029           IF ( oStoreObject.type = iTypeOfItemToDeliver) THEN
030           {
031             IF ( oStoreObject.stock <= iNumberOfItemsToDeliver)
032             {
033               iNumberOfItemsToDeliver:( iNumberOfItemsToDeliver -
                         oStoreObject.stocklevel )
034               oStoreObject.stock:( "" )
035             }
036             ELSE
037             {
038               oStoreObject.stock:(tokdiff (oStoreObject.stock,
                        tFIFOListOfItems(iNumberOfItemsToDeliver))
039 } } } } } } }
```

Nachdem der Simulationslauf beendet wurde, stehen die Analysefunktionalitäten zur Verfügung. Mittels der Analysefunktionalität „Timestamp“ kann eine beliebige Startperiode im simulierten Zeitrahmen ausgewählt werden. Die Animation stellt alle in dieser Periode aktiven dynamischen Objekte des Material- und Informationsflusses dar. Um die Synchronisation und Ablauffolge dieser Objekte studieren zu können, wird die Reihenfolge der Zustandsänderungen in einer Periode durch die Angabe von sogenannten Microsteps verdeutlicht. Der Benutzer der Animation hat die Möglichkeit über zwei pfeilförmige Icons in der Menüleiste, siehe Abbildung 70 rechts oben, die Simulationsuhr schrittweise, Periode für Periode, nach vor oder zurück zu stellen. Die Animationskomponente aktualisiert das Modell entsprechend der Simulationszeit und zeigt den neuen Zustand. Es kann somit einen guten Einblick in das Verhalten des Modells erlangt werden.

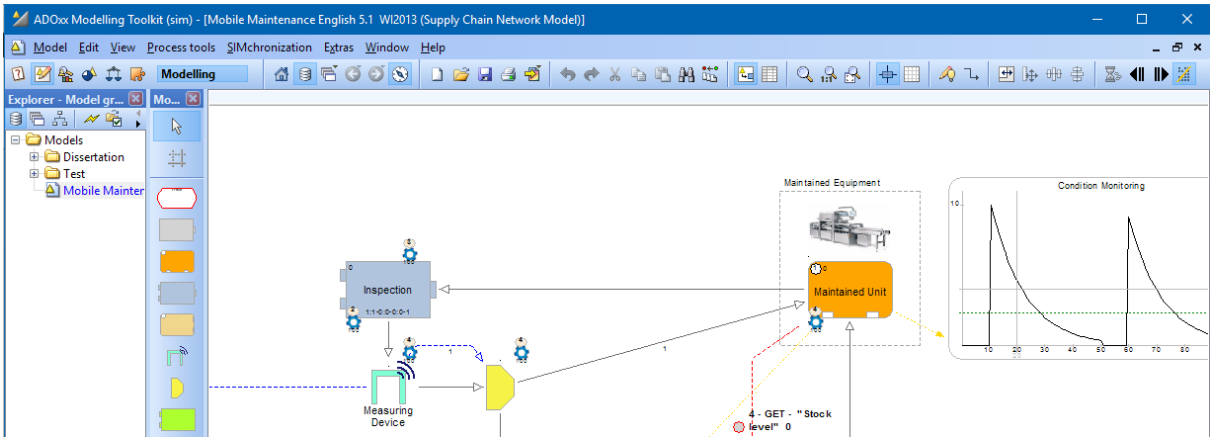


Abbildung 70: Icons in der Menüleiste zur Steuerung der Animation

Die zweite Analysefunktion fordert den Benutzer auf, die ID eines Items, also eines dynamischen Objekts des Materialflusses, einzugeben. Die Animation stellt darauf hin schrittweise oder in einem Durchlauf die „Lebensgeschichte“ des Items dar. Sollte das Item in der Supply Chain durch die Verwendung anderer Items produziert worden sein, so werden diese Items ebenfalls angezeigt, genauso wie Items, die aus dem zu analysierenden Item im weiteren Verlauf entstanden sind.

Da die Simulationsergebnisse in die Modelle zurückgeschrieben wurden, wird die Standardfunktionalität von ADOxx verwendet, um Analysen und Reports zu erzeugen.

8. Evaluation

Um die, in dieser Arbeit entwickelte Methode SIMchronization zu evaluieren, wurden in Kapitel 3 12 Anforderungen abgeleitet und definiert, welche die Methode erfüllen muss, um eine positive Antwort auf die Forschungsfrage zu sein. Im Folgenden werden anhand der prototypischen Implementierung einige praxisnahe und forschungsrelevante Beispiele besprochen und damit die Erfüllung von Anforderungen gezeigt.

8.1 Ersatzteileingang

Das erste Evaluierungsbeispiel bildet die Eingangsbearbeitung für Ersatzteile modellhaft ab. Wie in Abbildung 71 ersichtlich, startet der Materialfluss mit dem Beschaffungsvorgang der die Lieferung beinhaltet, die eintreffenden Ersatzteile werden in einem Zwischenlager gelagert, bis sie einer Qualitätskontrolle unterzogen werden. Diese Kontrolltätigkeit ist mittels des Make-Objekts abgebildet. Anschließend werden die Teile im Ersatzteillager gelagert bis sie vom Maintenance-Prozess, hier modelliert als Deliver-Objekt, verbraucht werden.

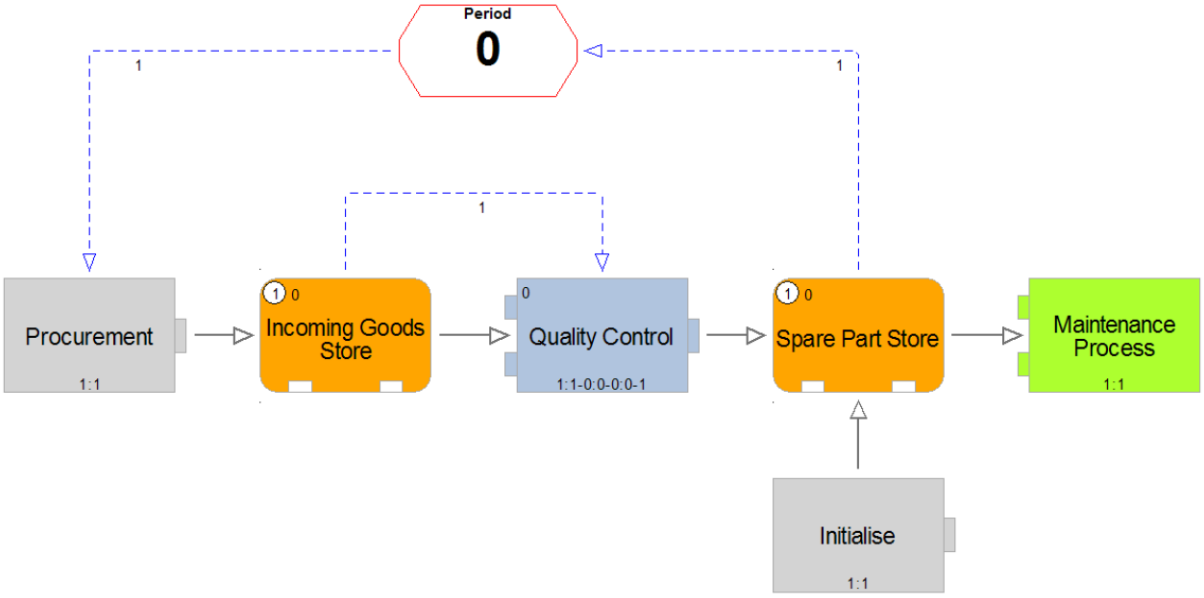


Abbildung 71: Beispielmodell Ersatzteileingang

Die Steuerung des Materialflusses erfolgt auf zweierlei Arten. Zentral durch das Plan-Objekt und dezentral durch das Make-Objekt. Die zentrale Steuerung liest jede zweite Periode den Ersatzteillagerbestand aus und bestellt genau die Anzahl von Teile, sodass der Ziellagerstand von 2 erreicht wird. Die Bestellung wird durch eine Message an das Source-Objekt ausgelöst.

Die Action-Regel des Plan-Objekts ist:

```
001 set( iUneven, ceil( TIMESTAMP / 2 ) - floor( TIMESTAMP / 2 )),
002 cond( iUneven = 0, (
003   set( iOrder, 2 - get(INFOIN1, "Stock level" )),
004     set ( send[1], STR INFOOUT1 + "#Order#" + STR iOrder )),
005   "")
```

Die dezentrale Steuerung des Make-Objekts ist sehr einfach, es frägt jede Periode den Lagerstand des zuliefernden Lagers ab, sobald Teile im Lager verfügbar sind, werden diese einer Qualitätsprüfung unterzogen. Die Regel im Order-Attribut des Make-Objekts ist:

```
001 get(INFOIN1, "Stock level")
```

Das Auftragsvolumen ergibt sich aus der Anzahl der gelagerten Teile im Store-Objekt, das über den eingehenden Information flow channel mit Index 1 verbunden ist.

Zur Erstellung eines anschaulichen Beispiels sind zwei weitere Ergänzungen notwendig. Die Initialisierung des Modells erfolgt mittels des Source-Objekts “Initialise” das in Periode 1 das Lager mit 2 Ersatzteilen füllt und es wurde dem Deliver-Objekt, welches den Ersatzteil-verbrauchenden Instandhaltungsprozess abbildet eine vordefinierte Verbrauchsfunktion vorgegeben. Die Order-Regel des Deliver-Objekts ist:

```
001 VAL token ( "0 0 1 0 0 0 1 0 1 1 1 0 1 1 2", TIMESTAMP -1)
```

Diese Regel liest sequentiell die Liste des Auftragsvolumina aus.

Um die Zusammenhänge innerhalb der Supply Chain zu verstehen, wird die Simulation über 10 Perioden hinweg ausgeführt. Wird anschließend die schrittweise Animation für die einzelnen Perioden durchgeführt, so erhält man die in Abbildung 72, als eine Art Bildergeschichte, dargestellten Zustandsfolgediagramme. Jedes Diagramm enthält die Zustandsänderungen innerhalb einer Periode. Diese werden sequentiell als sogenannte Microsteps durchnummeriert.

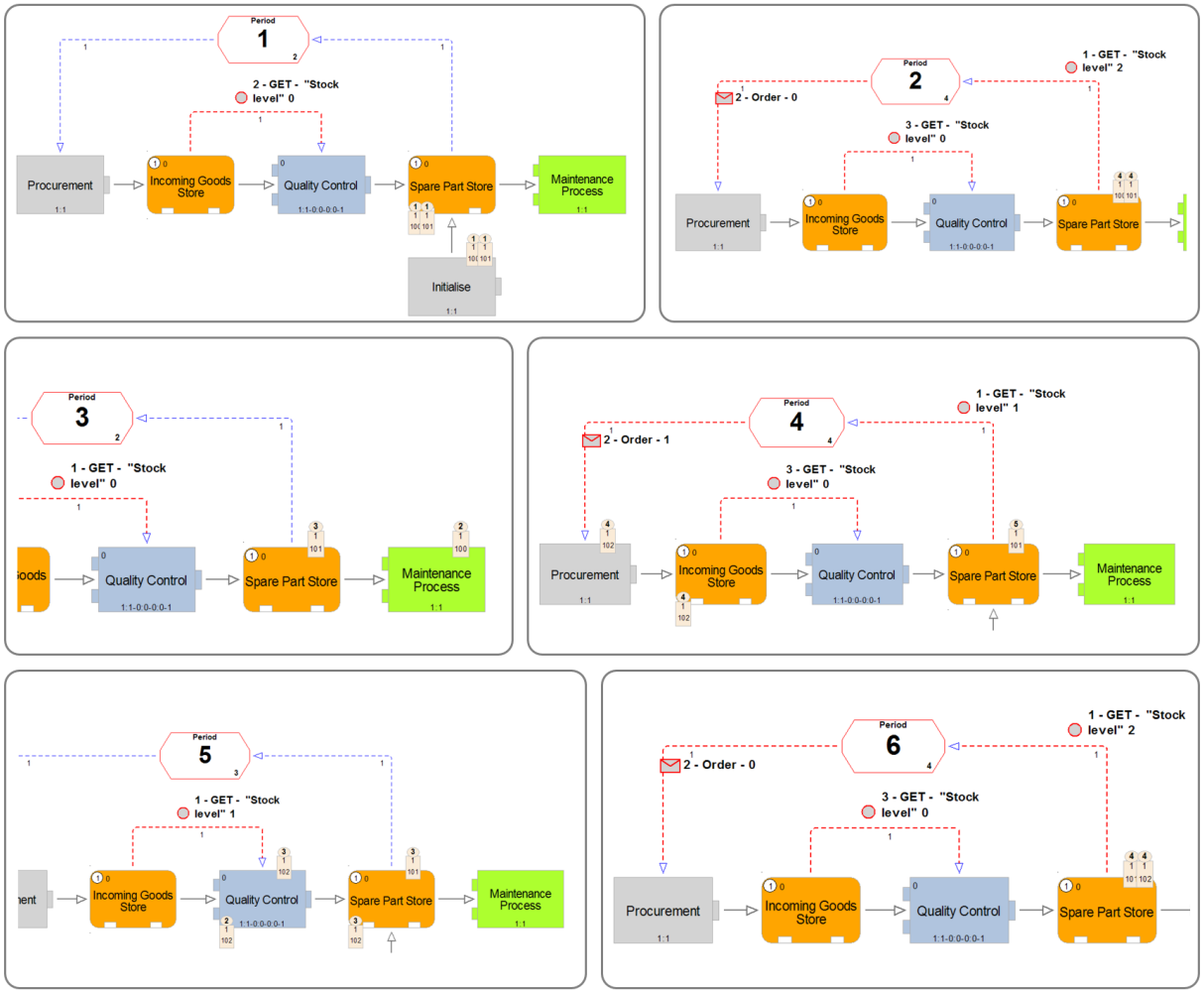


Abbildung 72: Zustandsfolgediagramme des Beispiels Ersatzteileingang

8.2 Information Sharing

Das folgende Anwendungsbeispiel der Methode SIMchronization zeigt anhand des Bullwhip-Effekts, einem klassischen Supply Chain Phänomen, wie die Methode dazu genutzt werden kann, verschiedene Arten des Informationsaustauschs zwischen Supply Chain Partnern zu analysieren und zu verbessern. Forrester (Forrester, 1969) beschreibt die natürliche Tendenz einer Supply Chain, Information über Kundenbedarfe zu verstärken und Informationsweitergabe zu verzögern und damit zu einem Aufschwingen der Nachfragekurve bei stromaufwärts gelegenen Supply Chain Partnern zu führen (Sahin und Robinson, 2002). Dieser Effekt wird Bullwhip Effekt (Lee et al., 1997) oder auch Forrester Effekt genannt. Mittels Modellierung der Informations- und Materialflüsse einer mehrstufigen Supply Chain soll in diesem Beispiel der Effekt herbeigeführt werden und

gezeigt werden, dass ein weitergehender Informationsaustausch zu einer besseren Synchronisation und damit Performanz der Supply Chain führt.

Das in Abbildung 73 enthaltene Modell stellt eine Supply Chain dar, deren Zweck die Belieferung von Endkunden mit einem Gut ist. Dafür arbeiten ein Supermarkt, ein Großhändler, Distributor und Rohstoff-Erzeuger zusammen. Das Deliver-Objekt bildet das Verhalten der Endkunden im Supermarkt ab, welche eine konstante, aber leicht schwankende Nachfrage zeigen. Die Partner der Supply Chain werden jeweils mittels eines Store-Objekts, das deren Lager abbildet, und eines Make-Prozesses abgebildet. Die verhaltensdefinierende Regel im Make-Objekt enthält die Bestellpolitik, die angewendet wird um Güter, von den, im Materialfluss stromaufwärts gelegenen Lieferanten zu erhalten. Die Source-Objekte sind Hilfsobjekte und dienen zur Initialisierung des Modells am Start eines Simulationslaufs.

Die Produktionsprogramme der Make-Objekte sind:

Tabelle 19: Produktionsprogramme der Make-Objekte des Beispiels

Make-Objekt	Bearbeitete Teile		Verbrauchte Teile		Erzeugte Teile		Bearbeitungszeit
	Anzahl	Typ	Anzahl	Typ	Anzahl	Typ	
Supermarket	1	1	0	0	0	0	**1**
Wholesaler	1	1	0	0	0	0	**2**
Distributor	1	1	0	0	0	0	**3**

Die bearbeiteten Teile werden nur durchgereicht, dies allerdings nicht in jeder Periode, sondern je nach Stufe in der Supply Chain nur jede zweite oder dritte Periode.

Um eine bessere Vergleichbarkeit der Ergebnisse in diesem Beispiel zu erzielen, verfolgen alle Supply Chain Partner über dieselbe Bestellpolitik. Um die Bestellmenge ihres Kunden in der nächsten Periode zu prognostizieren, betrachten sie die Bestellungen der letzten beiden Perioden und extrapolieren diese gemäß der im Folgenden dargestellten Formel.

$b_t \dots Bedarf\ zum\ Zeitpunkt\ t$

$l_t \dots Lagerstand\ zum\ Zeitpunkt\ t$

$z \dots Ziellagerstand$

$o \dots Produktionsmenge\ (Order)$

$s \dots gewichtete\ Entwicklung\ der\ Bedarfskurve$

$$s = (b_t - b_{t-2}) * 0.8 + (b_t - b_{t-1}) * 0.2$$

$$s = \begin{cases} s, & s \geq 0 \\ 0, & s < 0 \end{cases}$$

$$o = z + CEIL\ (s) - l$$

Wobei CEIL() positive Zahlen zur nächst höheren Ganzzahl aufrundet.

Diese Formel umgesetzt in die SIMchronization-Regelsprache führt zu folgendem Regelset in einem Make-Objekt:

```
001 set( sDelHistory, get( INFOIN2, "Delivery history - to be" )),
002 set( iStockLevel, get( INFOIN1, "Stock level" )),
003 set( iNrPeriods, tokcnt( sDelHistory ) - 1),
004 cond( TIMESTAMP >= 3, (
005   set( iLong, 0.8 * ( VAL token( sDelHistory, iNrPeriods ) -
        VAL token( sDelHistory, iNrPeriods - 2))),
006   set( iShort, 0.2 * ( VAL token( sDelHistory, iNrPeriods ) -
        VAL token( sDelHistory, iNrPeriods – 1 ))),
007   set( iCurrent, VAL token( sDelHistory, iNrPeriods )),
008   set( iMinumumStock, 15),
009   set( iOrder, iMinumumStock + ( iCurrent + iLong + iShort ) -
        iStockLevel ),
010   cond( iOrder < 0, 0, CEIL( iOrder ))
011 ),
012 0 )
```

Die Information über den eigenen Bedarf erhält eine Stufe in der in Abbildung 73 dargestellten Supply Chain Konfiguration über Beobachtung des eigenen Bestelleingangs. Die daraus abgeleitete Höhe der Bestellung an den eigenen Lieferanten ergibt sich aus der Abschätzung des Bedarfs in den nächsten Perioden. Die Simulation des Beispiels zeigt, dass die Nachfrage der Kunden im Supermarkt konstant um den Mittelwert 5 schwankt[76]. Der Lagerstand des Supermarkts folgt dieser Nachfrageschwankung. Im Diagramm ist erkennbar, dass der Lagerstand etwas stärker fluktuiert als die Kundennachfrage, der mittlere Lagerbestand ist dabei 7. Dieser Trend verstärkt sich in den stromaufwärts gelegenen Stufen der Supply Chain.

Diese Konfiguration der Supply Chain ist unter anderem in sehr wettbewerbsorientierten Märkten zu finden, in denen die einzelnen Supply Chain Partner jederzeit damit rechnen müssen, gegen andere, wettbewerbsfähigere Lieferanten ausgetauscht zu werden. Dadurch ist die Bereitschaft zum Informationsaustausch, unter anderem wegen dem Risiko, dass diese Information ein Konkurrent enthält, gering.

Die zweite Konfiguration, dargestellt im Modell Abbildung 74 enthält dieselben Supply Chain Partner, Bestellpolitiken, und Materialflusskonfiguration, als das eben diskutierte Beispiel.

[76] Die horizontale gestrichelte Linie in den Diagrammen zeigt den Mittelwert der dargestellten Simulationsergebnisse dar

Einziger Unterschied liegt in der Konfiguration des Informationsflusses. Nun können alle Supply Chain Partner direkt auf die Information über die Nachfrage des Endkunden

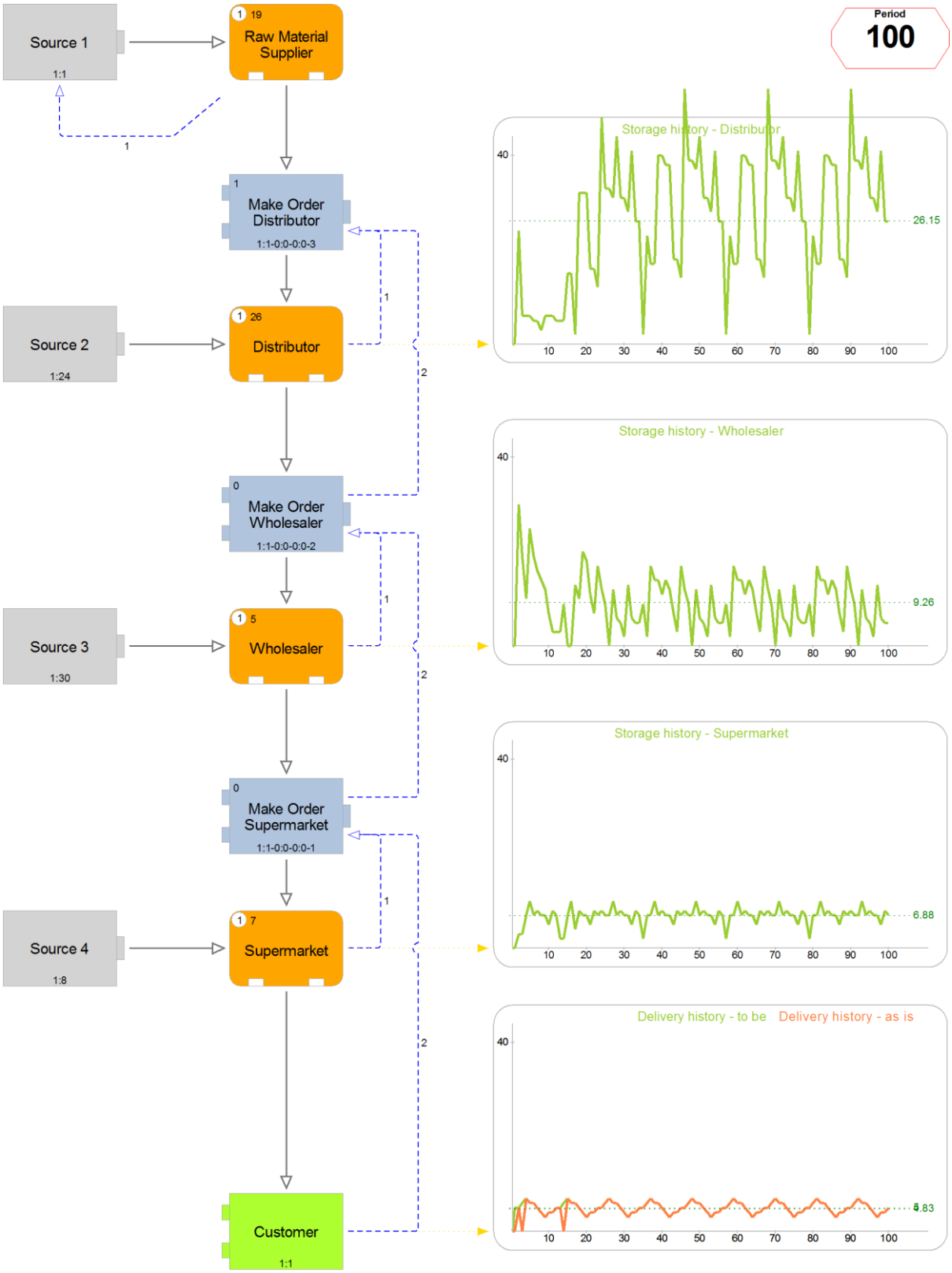


Abbildung 73: Simulationsergebnisse ohne Information Sharing

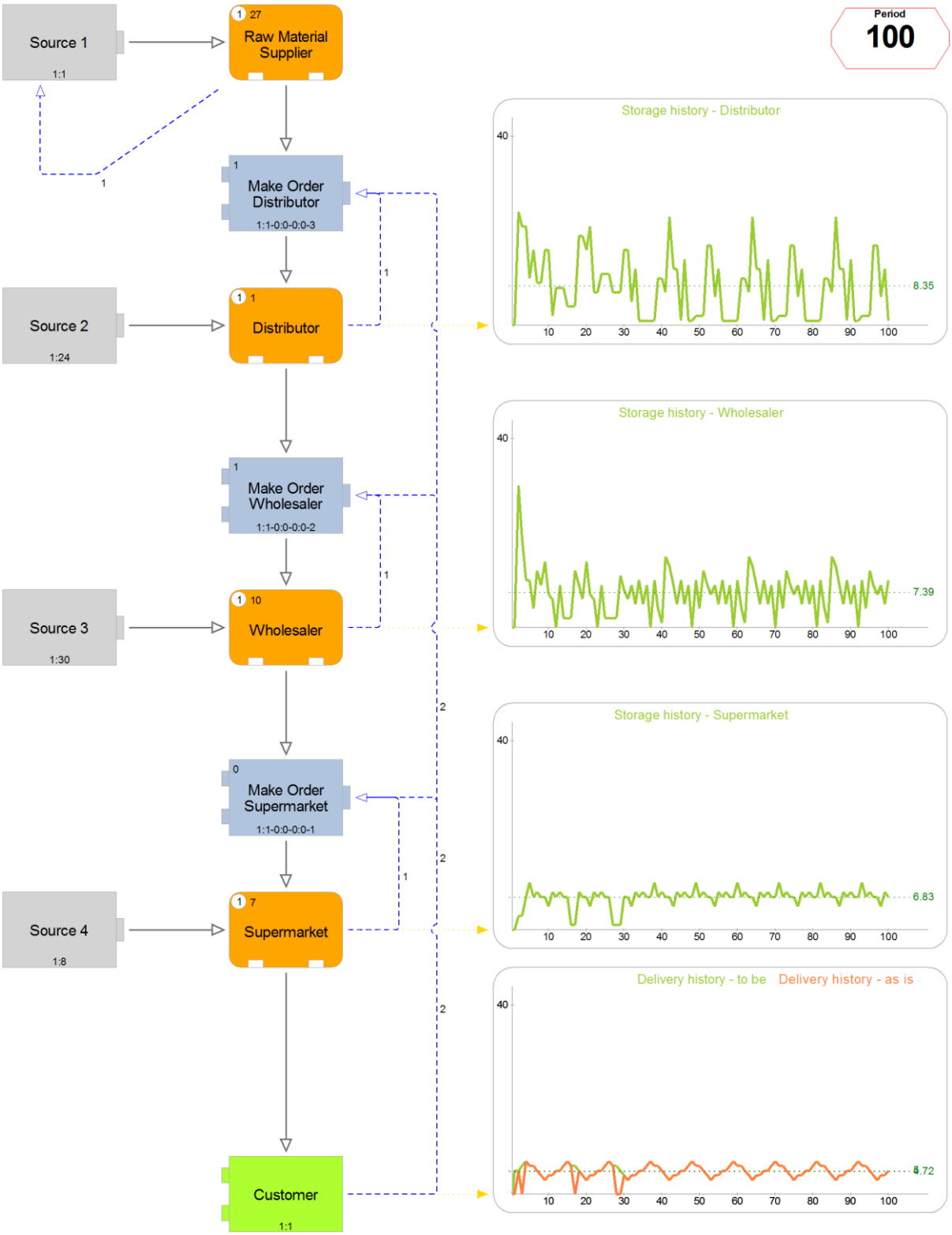


Abbildung 74: Simulationsergebnisse mit Information Sharing

zugreifen. Es kommt also zu keiner Verzögerung in der Informationsweitergabe zwischen den einzelnen Stufen. Wie aus den Diagrammen ersichtlich, ist der Unterschied für die Lagerstandentwicklung des Supermarktes nicht erheblich. Er ist weiterhin in der Lage die

Kundennachfrage zu einem großen Anteil zu befriedigen. Auswirkungen der Informationsweitergabe sind beim Großhändler und vor allem beim Distributor zu sehen. Ein Vergleich der durchschnittlichen Lagerstände findet sich in Tabelle 20.

Tabelle 20: Durchschnittliche Lagerstände im Beispiel Information Sharing

Store-Objekt	Durchschnittlicher Lagerstand	
	Ohne Information-Sharing	Mit Information-Sharing
Supermarket	7	7
Wholesaler	9	7
Distributor	26	8
Summe	42	22

8.3 Instandhaltungs-Supply Chain

Das Analyseobjekt in diesem Beispiel, siehe Abbildung 75, ist eine Instandhaltungs- oder Maintenance Supply Chain, die regelmäßig den Zustand eines Teils einer instand zu haltenden Anlage inspiziert. (Prackwieser, 2013) Falls eine, den Zustand des Objekts repräsentierende Kenngröße unter ein definiertes Limit fällt, wird der Austausch dieses Teils veranlasst und gegebenenfalls dabei ein Ersatzteil bestellt. Ziel der Analyse ist, die Anlagenausfallszeit durch eine effiziente Inspektionsplanung zu verringern und dabei gleichzeitig die notwendige Ersatzteil-Lagermenge im eigenen Lager zu verringern.

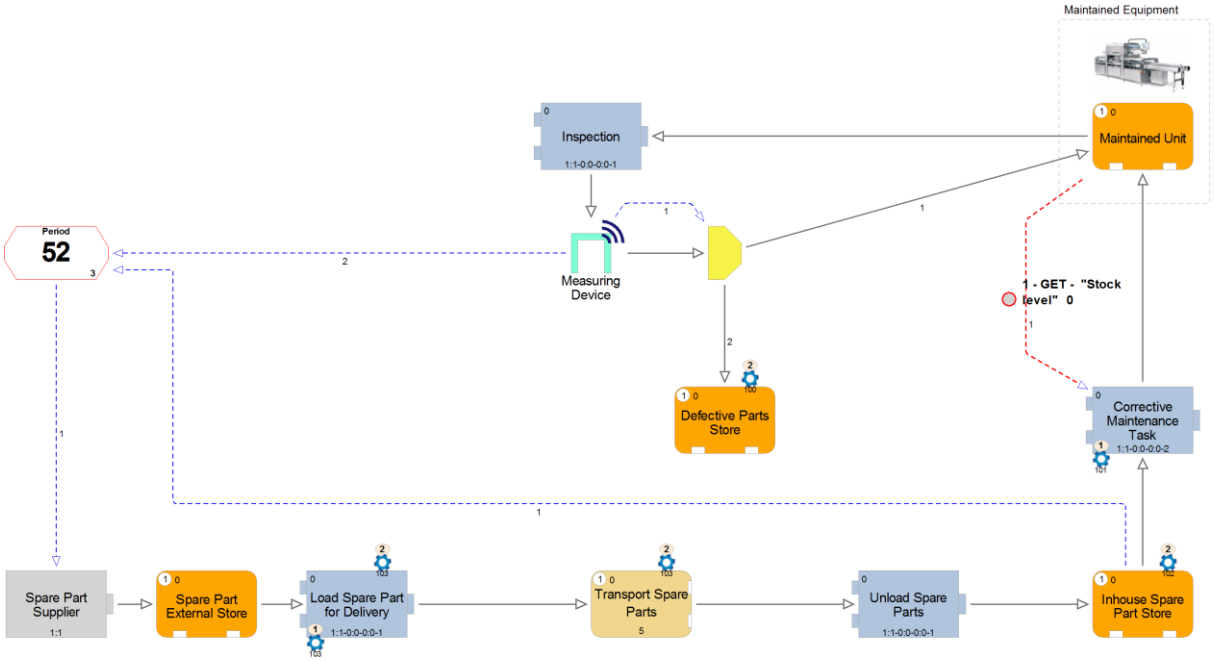


Abbildung 75: Beispielmodell Instandhaltungs-Supply Chain

Die instand zuhaltende Anlage ist durch die Aggregation im rechten oberen Eck des Modells dargestellt. Die Einbindung eines Fotos der Anlage verbessert die Wiedererkennung durch den Modellbetrachter. Das Store-Objekt stellt die Lokation der zu wartenden Einheit dar. Das im Store-Objekt gelagerte Objekt, ist somit das zu wartende Teil das in dieser Lokation eine Zustandsänderung, beispielsweise durch Abnutzung, erfährt. Diese Zustandsänderung wird definiert durch die folgende Regel:

```
001 cond( type( READ[5] ) = "undefined",
002   set( WRITE[5], READ[3] ), "" ),
003   set( WRITE[4], 10 * pow( 0.9, ( TIMESTAMP - READ[5] ) * 0.8 ))
```

READ [3] liest dabei den Einlagerungszeitpunkt des Teils aus, der in dem Szenario dem Installationsdatum des Teils entspricht. Sein aktueller Zustand wird in den 4. Speicherplatz des Teils geschrieben. Das Make-Objekt „Inspection“ bildet die Durchführung der Inspektion ab, also den Ausbau, Messung und Wiedereinbau des zu inspizierenden Teils. Die Regel in seinem Order-Attribut legt fest, dass eine Inspektion alle 5 Perioden zu erfolgen hat.

```
001 cond( round( TIMESTAMP / 5 ) = ( TIMESTAMP / 5 ), 1, 0 )
```

Die eigentliche Bewertung des Zustands erfolgt im Reader-Objekt, das den Zustandswert im Speicherplatz 4 des Objekts ausliest. Der dazu notwendige Befehl wird im Attribut Action des Reader-Objekts hinterlegt:

```
001 STR( READ[4] )
```

Die zugehörige Reaction-Regel des Reader Attributs nutzt das Ergebnis der in Action hinterlegten Regel, das in der Variable RESULT gespeichert ist.

```
001 set( send[1], STR INFOOUT1 + "##PART-" +
002 STR cond( VAL RESULT <= 0.4,
003   ( set( send[2], STR INFOOUT2 + "#Replace#1" ),
004     2 ),
005   1
006 ))
```

Der Reader sendet an das Switch-Objekt eine Nachricht, deren Inhalt bestimmt, ob das Teil durch den Material flow channel mit dem Index 1 oder 2 bewegt werden soll, entscheidend dabei ist, ob das Messergebnis für den Zustand des Teils größer als 0,4 ist. Pfad 1 führt zum Wiedereinbau in die Anlage, während Pfad 2 das Teil im Lager für defekte Teile ablegt und das Make-Objekt „Corrective Maintenance Task“ baut, sofern verfügbar, ein neues Teil in 2 Perioden ein. Wird das Teil ausgeschieden, so wird zusätzlich vom Reader-Objekt eine „Replace“ Nachricht an das Plan-Objekt gesendet. Die Steuerung der Ersatzteilbeschaffung erfolgt durch ein zentrales Bestellsystem. Der Transport vom Lieferanten ins Haus dauert 5 Perioden. Um die Methode zu evaluieren, werden die Auswirkungen von drei verschiedenen Ersatzteilpolitiken analysiert und die Elastizität der Supply Chain in Bezug auf ungeplante Störungen getestet.

Das Plan-Objekt löst in jedem Fall eine Bestellung eines Ersatzteils durch das Deliver-Objekt aus, sobald ein Teil in der zu wartenden Einheit ausgetauscht wird und damit eine Nachricht mit der Message-ID „Replace“ eintrifft. Diese wird durch folgende Regel im Reaction-Attribut des Plan-Objekts veranlasst:

```
001 cond( MESSAGEID = "Replace",
002   set( send[2], STR INFOOUT1 + "#Order#1" ),
003   ""
004 )
```

Zusätzlich zu dieser Regel erfolgt eine aktive Prüfung des Ersatzteillagerstands durch das Plan-Objekt.

```
001 cond((( TIMESTAMP / 5 ) = round( TIMESTAMP / 5 )) OR TIMESTAMP = 1,
002   cond( get(INFOIN1, "Stock level" ) <=1,
003     set( send[2], STR INFOOUT1 +"#Order#1" ),
004   "" ),
005 "")
```

Durch die Simulation wird die Lagerstandentwicklung des werksinternen Ersatzteillagers studiert und dabei betrachtet, wie sich die Anlagenverfügbarkeit entwickelt. Die Anlagenverfügbarkeit ist binär, entweder 0 oder 1. Während der geplanten Inspektion eines Teils ist die Anlage nicht verfügbar. Es wurden drei Szenarien untersucht, die Ergebnisse der Analyse sind in Abbildung 76 dargestellt.

- Im ersten Szenario wird alle 5 Perioden der Lagerstand des In-House Ersatzteillagers geprüft und sobald der Lagerstand unter 2 fällt, wird 1 Ersatzteil bestellt.
- Im zweiten Szenario wird alle 10 Perioden der Lagerstand des In-House Ersatzteillagers geprüft und sobald der Lagerstand unter 2 fällt, wird 1 Ersatzteil bestellt.
- Im dritten Szenario wird alle 10 Perioden der Lagerstand des In-House Ersatzteillagers geprüft und sobald der Lagerstand 0 ist, wird 1 Ersatzteil bestellt.

Die Simulation des dritten Szenarios ergibt den geringsten notwendigen Lagerbedarf bei gleichbleibender Anlagenverfügbarkeit.

Das vierte Szenario basiert auf dem dritten Szenario, fügt aber einen ungeplanten Ausfall des Teils hinzu, um zu prüfen, ob die Supply Chain elastisch genug ist, um auf diese Störung schnell zu regieren. Der Ausfall wird abgebildet in dem der Zustand des Teils in Periode 54 bis 59 auf 0 gesetzt wird. Um die Untersuchung realitätsnäher zu gestalten, sollte der Eintrittszeitpunkt diese Störung einer Zufallsverteilung unterliegen. In dem Beispiel wurde aus Gründen der Nachvollziehbarkeit darauf verzichtet.

Interessantes Ergebnis dieser Störung ist, dass sich nicht nur die Anlagenverfügbarkeit verschlechtert, sondern durch die spezielle Abfolge von Material- und Informationsflüssen auch der durchschnittliche Lagerbestand höher ist als in Szenario 3.

Abbildung 76: Simulationsergebnisse des Beispiels Instandhaltungs-Supply Chain

Eine weitergehende Analyse des Modells zeigt, dass der nicht verwendete Ersatzteil-Lagerbestand hauptsächlich durch ein Synchronisationsproblem von Abfragezeitpunkten des hausinternen Lagerstands und der durch die Dauer des Transports verursachten Zeitverzögerung zwischen Bestellzeitpunkt und Lagereingang entsteht. Passt man die Abfragefrequenz des Lagerstands an die Lieferzeit an und simuliert das Modell erneut, so reduziert sich bei gleichbleibender Anlagenverfügbarkeit die durchschnittliche hausinterne Ersatzteil-Lagermenge um 1 Stück.

8.4 Radio Frequency Identification - RFID

Mit diesem Beispiel, vergleiche auch (Prackwieser, 2014), soll die Anwendbarkeit der Methode für die Analyse von Auto-ID-gestützten Supply Chains gezeigt werden. Neben Barcodes gewinnt insbesondere die RFID-Technologie in der Instandhaltung immer mehr an Bedeutung, weshalb die Abbildung und Analyse der sich daraus ergebenden Informationsflüsse besonders interessant ist.

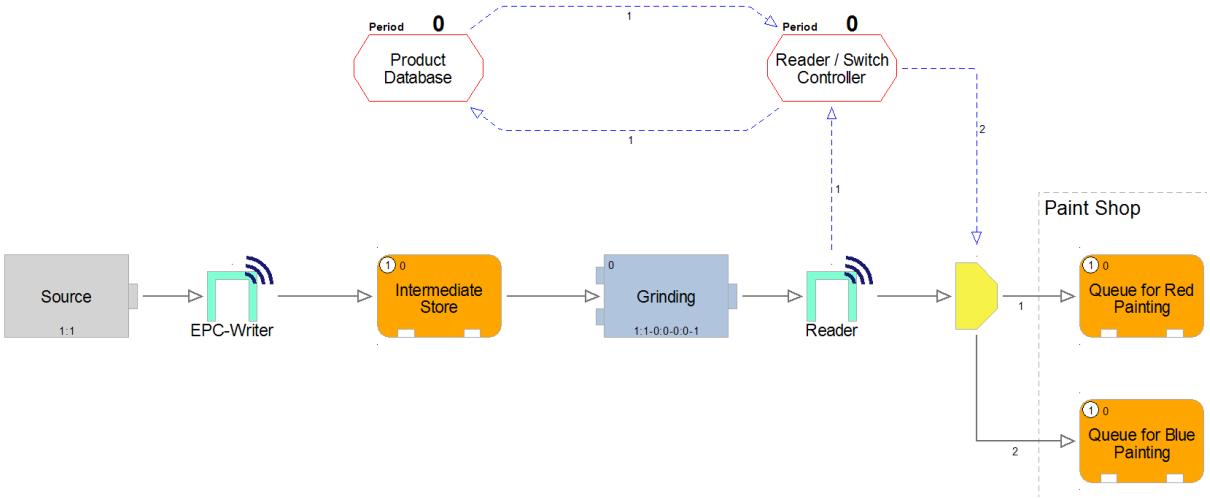


Abbildung 77: Beispielmodell zur Radio Frequency Identification – RFID

Das in Abbildung 77 dargestellte Modell bildet einen Ausschnitt eines Fertigungsprozesses ab, in dem Teile geschliffen und je nach ID, vorbestimmt, entweder mit roter oder mit blauer Farbe lackiert werden. Der Materialfluss beginnt mit einem Source-Objekt, das als Platzhalter für den vorgelagerten Prozess dient, der in dieser Analyse nicht von Interesse ist. Das Source-Objekt generiert Teile, die sofort ein Reader-Objekt durchlaufen, welches einen teilespezifischen Code (EPC) auf den Smart-Tag schreibt. Der EPC besteht aus der Produktklassen-spezifischen Nummer „9037“ und einer teileindividuellen Nummer. Die im Action-Attribut des Objekts EPC-Writer hinterlegte Regel lautet:

```
001 set( WRITE[4], "9037" + STR PART)
```

Im Speicherplatz 4 jedes Teils wird der Code als String hinterlegt. Die Variable PART stellt die ID des dynamischen Objekts zur Verfügung.

Die Teile werden im Zwischenlager gelagert und anschließend im Make-Objekt geschliffen, laut Produktionsprogramm dauert dies 1 Periode. Um die Teile nach der Schleiferei in die richtige Position zum anschließenden Lackieren zu bewegen, müssen die Teile gemäß ihres EPC Codes gelenkt werden.

Dafür laufen die dynamischen Objekte des Materialflusses zunächst wieder durch ein Reader-Objekt, dessen Action-Attribut folgende Regel enthält:

```
001 cond( type( READ[4] ) = "undefined",
002   "",
003   set( send[1], STR INFOOUT1 + "#EPC#" + READ[4] ))
```

Eine Ausführung dieser Regel führt dazu, dass an das Objekt, das am anderen Ende des Information flow channels mit Index 1 ist, eine Nachricht gesendet wird, welche den Nachrichten Identifier „EPC“ und die EPC Nummer als Nachrichteninhalt hat. Die Nachricht wird an das Plan-Objekt Reader / Switch Controller gesendet und wie jede eingehende Nachricht von der, im Attribut Reaction stehenden Regel ausgewertet.

```
001 cond(MESSAGEID = "EPC",
002   set(send[1], STR INFOOUT1 + "#Request#" + MESSAGESTR),
003   ""),
004 cond(MESSAGEID = "Reply",
005   cond(token(MESSAGESTR,1,"-") = "red",
006     set(send[0],STR INFOOUT2 + "##" + token(MESSAGESTR,0,"-") + "-1"),
007     set(send[0],STR INFOOUT2 + "##" + token(MESSAGESTR,0,"-") + "-2")),
008   "")
```

Für die Kommunikation mit dem Reader ist nur der erste Teil des Regelsets relevant, da diese Regel auf Nachrichten mit dem Identifier „EPC“ reagiert. Es wird eine Nachricht „Request“ über den ausgehenden Information flow channel 1 an das Plan-Objekt Product Database gesendet. Inhalt der Nachricht ist der EPC.

Die Reaction-Regel dieses Plan-Objekts lautet:

```
001 cond( MESSAGEID = "Request",(
002   set ( iEpcId, VAL MESSAGESTR ),
003   set ( mProductData, VAL aval( "Description" )),
004   set ( sColor, mProductData[ iEpcId ]),
005   set ( send[1], STR INFOOUT1 + "#Reply#" + copy ( MESSAGESTR, 4, -1) +
   "-" + sColor)),
006   "")
```

Die produkt- und teilespezifischen Daten werden aus einer Liste ausgelesen, die in diesem Beispiel im Description-Attribut des Plan-Objekts hinterlegt ist, die Liste hat das ADOxx-eigene map-Datenformat:

```
001 {
002 9037100:"red",
003 9037101:"blue",
004 9037102:"red",
005 9037103:"blue",
006 9037104:"blue",
007 9037105:"red",
008 9037106:"blue",
009 9037107:"red",
010 9037108:"red",
011 9037109:"red",
012 9037110:"blue",
013 9037111:"blue",
014 9037112:"red",
015 9037113:"red",
016 …
017 }
```

Die Abarbeitung der Regel endet mit dem Versand einer Antwort „Reply“ an das Plan-Objekt Reader / Switch Controller, welches die korrekte Farbe für das Teil beinhaltet. Der Reader / Switch Controller setzt im zweiten Teil des oben angeführten Regelsets die Information über die richtige Farbe in eine Nachricht um, die den Switch anweist, das Teil entweder über den aus dem Switch ausgehenden Material flow channel mit Index 1 oder den, mit Index 2 zu bewegen.

Tabelle 21 zeigt das Modell in Periode 15 und alle Bewegungen der dynamischen Objekte des Material- und Informationsflusses sequentiell nummeriert in Microsteps.

Tabelle 21: Microsteps der Periode 15 des Beispiels RFID

Microstep	Statisches Ausgangsobjekt	Statisches Zielobjekt	Dynamisches Objekt
2	Grinding	Reader	Teil 113
3	Reader	Reader / Switch Controller	Message „EPC - 9037113“
4	Reader / Switch Controller	Product Database	Message „Request - 9037113“
5	Product Database	Reader / Switch Controller	Message „Reply - 113-red“
6	Reader / Switch Controller	Switch	Message „113-1“
7	Switch	Queue for Red Painting	Teil 113

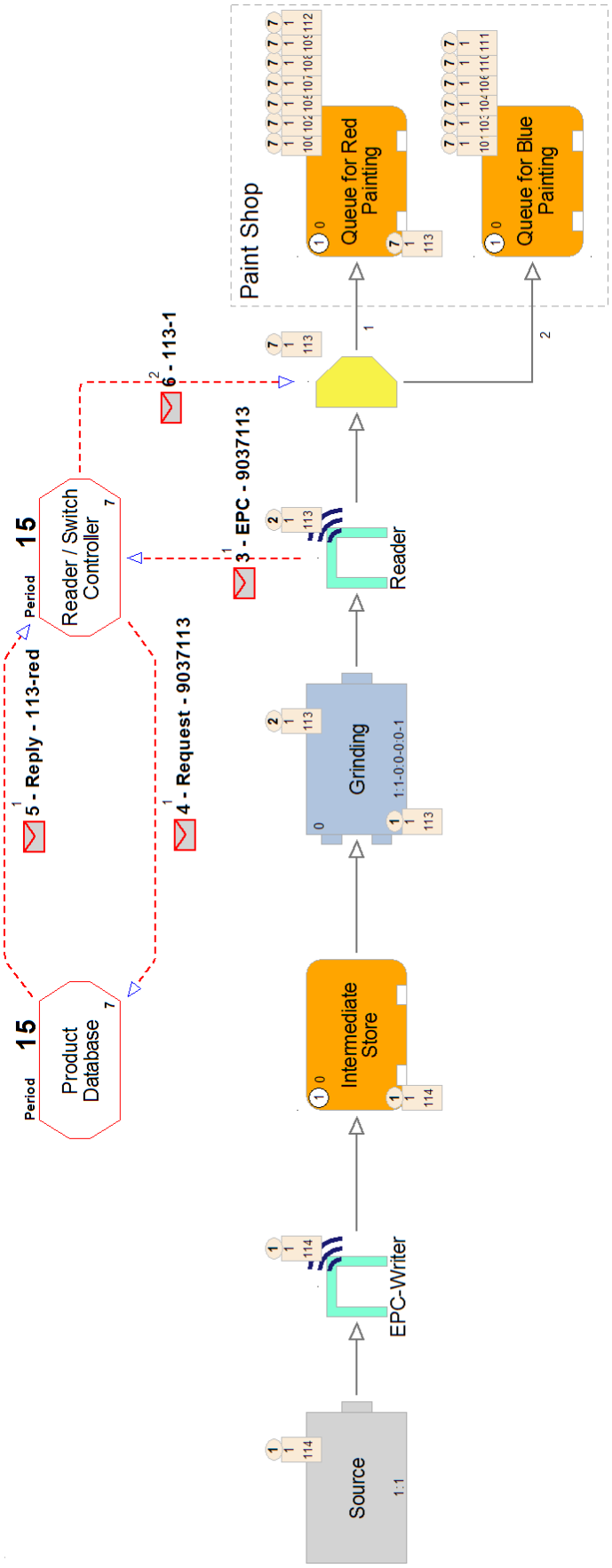


Abbildung 78: Zustandsfolgediagramm aus Beispiel RFID

Wie aus Abbildung 78 ersichtlich, sind folgende Teile für die rote und blaue Lackierung vorgesehen und in dem jeweiligen Store-Objekt gelagert:

Rot: 100, 102, 105, 107, 108, 109, 112, 113

Blau: 101,103, 104, 106, 110, 111

Dieses Ergebnis entspricht der Vorgabe aus der Product Database.

8.5 Fertigungslinie

Dieses Beispiel wird nicht im Detail beschrieben, sondern es soll demonstrieren, dass auch größere Modelle mit der Methode abgebildet und analysiert werden können. Abbildung 79 zeigt das Modell, das eine high-level Automobilproduktions-Supply Chain darstellt. Wie alle anderen Beispielmodelle ist es im Lieferumfang der Methode SIMchronization enthalten, die mittels der Plattform Open Models Laboratory (OMiLAB) zur Verfügung gestellt wird.

Der obere Bereich des Modells stellt die Lieferkette für Reifen dar. Jeweils vier Reifen werden in einen Container verpackt, mittels des Transport-Objekts versendet und anschließend wieder entpackt, bevor sie am Automobil montiert werden. Der untere Materialfluss beginnt mit einem Source-Objekt, das dezentral gesteuert, über einen Abruf das nachfolgende Store-Objekt abfrägt und je nach Lagerstand eine Bestellung für Bleche auslöst. Im nachfolgenden Presswerk werden die Rohkarossen geformt und je nach Auftrag vom zentralen Plan-Objekt in rot, grün oder blau lackiert. Das Plan-Objekt frägt dazu die Lagerstände der Auslieferungslager ab und sendet je nach Ergebnis einen Auftrag zur Änderung des Produktionsprogramms an das Make-Objekt. Rote Autos bekommen über einen zusätzlichen Identifikations-, Beschaffungs- und Produktionsschritt noch einen Gepäckträger montiert. Nach diesem Fertigungsschritt wird in die lackierte Karosse der Motor verbaut, die Reifen montiert und der Wagen in das jeweilige Auslieferungslager gebracht. Abbildung 79 zeigt den Stand des Modells in der Periode 51. Die Diagramme geben den Verlauf der Lagerstände über die gesamte zuvor simulierte Zeitdauer von 100 Perioden wieder.

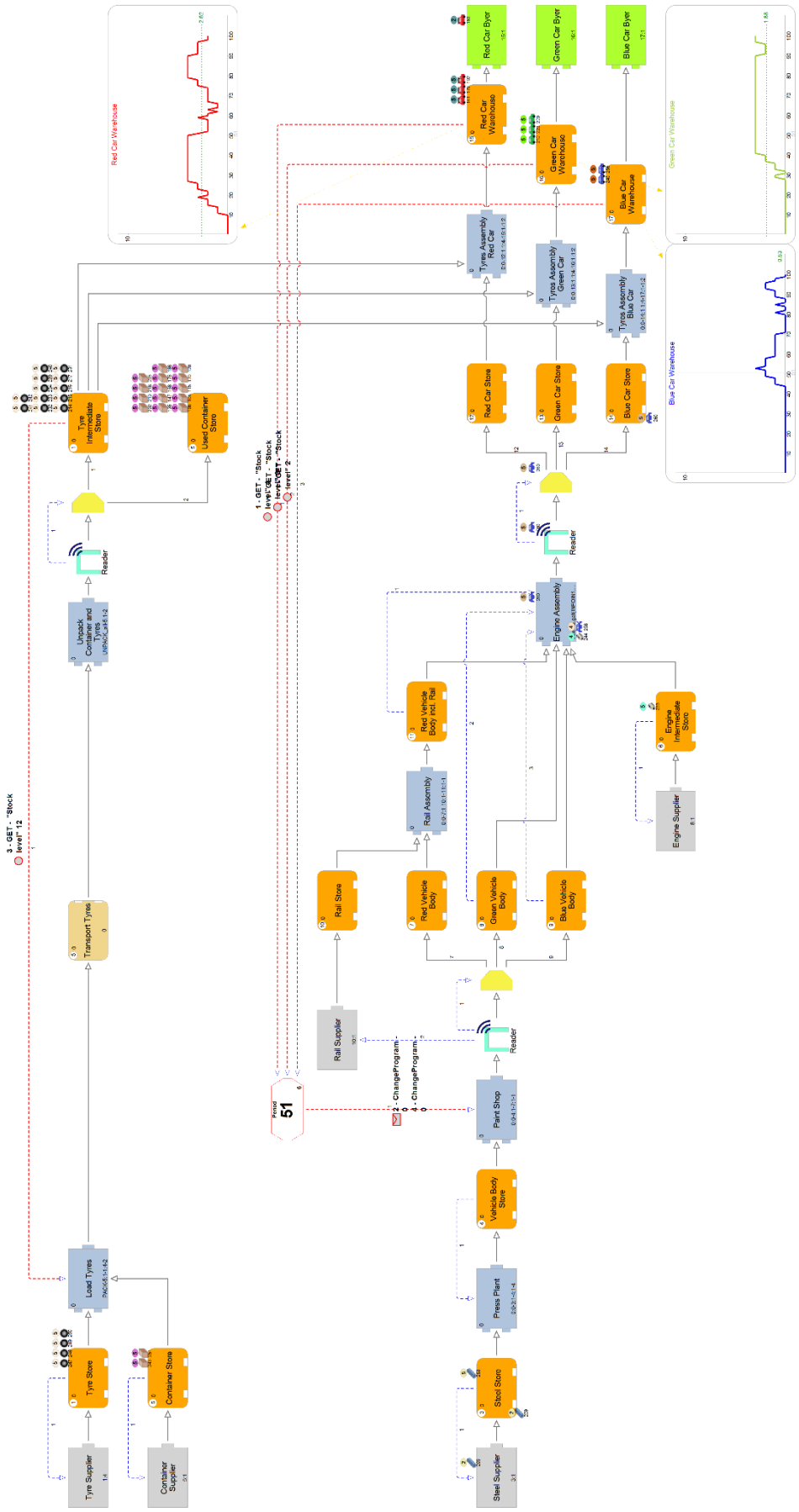


Abbildung 79: Beispielmodell Fertigungslinie

8.6 Abgleich mit den Anforderungen

Auf Basis der Ergebnisse der Metamodelldefinition, Implementierung und der oben dargestellten Anwendungsbeispiele wird nun die Methode anhand eines Abgleichs mit den, in Kapitel 3 aufgestellten Anforderungen evaluiert.

Anforderung 1: Es ist ein Metamodell bereitzustellen und Notationsregeln zu definieren, nach denen ein formal korrektes Modell erstellt werden kann.

Es wurde ein Metamodell und darauf gültige Notationsregeln für die Methode SIMchronization definiert und dessen Herleitung dargestellt. Das Metamodell wurde mittels FDMM formal beschrieben und durch die Implementierung auf der Plattform ADOxx konnte gezeigt werden, dass mit dem Metamodell formal korrekte Modelle erstellt werden können, auf die der Simulationsalgorithmus fehlerfrei anwendbar ist.

Anforderung 2: Die Modellierungssprache muss mehrere Ebenen von Abstraktion und die Wiederverwendung von Modellinhalten unterstützen.

Das in dieser Arbeit entworfene konzeptionelle Modell sieht in einer Erweiterung die Referenzierung von Supply Chain Network Models aus Make-Objekten vor. Es wurde allerdings darauf verzichtet, diese Funktionalität im Prototyp umzusetzen.

Anforderung 3: Die Modellierungssprache muss erweiterbar sein und auf verschiedene Anwendungsszenarien hin anpassbar sein.

Durch die Entscheidung diese Methode auf der Metamodellierungsplattform ADOxx zu implementieren, ist diese Anforderung erfüllt. Zum einen kann ein Metamodellierer weitere Klassen oder Attribute dem Metamodell in ADOxx hinzufügen, zum anderen sind alle Mechanismen und Algorithmen in einer offenen Skriptsprache verfasst, die eine Änderung oder das Hinzufügen von Funktionalitäten erlaubt.

Anforderung 4: Die Modellierungssprache muss über eine einfach verständliche und ausdrucksstarke Notation mit domänenspezifischer Begrifflichkeit verfügen.

Ein wichtiges Entscheidungskriterium beim Entwurf der Modellierungssprache war, dass der Umfang, der zur Verfügung stehenden Objekttypen überschaubar bleibt, aber zugleich alle in der Domäne Maintenance Supply Chain auftretenden Typen von Prozessschritten mittels spezifischer Klassen abbildbar sind. So wurden viele Begriffe und Konzepte aus bekannten und etablierten Standards, wie dem SCOR-Modell übernommen. Sehr domänenspezifische Konzepte, wie die Zustandsveränderung von instand zu haltenden Teilen wurden der Methode hinzugefügt.

Anforderung 5: Alle Bedienungselemente der Methode sollten in eine Anwendungsoberfläche integriert sein.

Auch wenn es Argumente gibt, die einzelnen Funktionalitäten auf spezialisierte Tools zu verteilen, so hat die Arbeit mit dem Prototyp gezeigt, wie benutzerfreundlich und effektiv die Modellierung und Analyse von statten geht, wenn sowohl Modelle als auch Auswertungsfunktionalitäten in einem Tool und einer Benutzeroberfläche integriert sind.

Anforderung 6: In der Maintenance Supply Chain bearbeitete und transportierte materielle Teile und übertragene Informationen müssen im Modell individuell beschreibbar und auswertbar sein, um die dynamischen Zusammenhänge und Abhängigkeiten, zwischen Informations- und Materialfluss erkennen und deren Synchronisation verbessern zu können.

Wie aus den Anwendungsbeispielen ersichtlich, ist diese Anforderung erfüllt.

Anforderung 7: Die Modellierungsmethode muss die Erstellung und Analyse von Modellen mit zentraler, dezentraler und autonomer Supply Chain Steuerung inklusive Auto-ID Verfahren unterstützen.

Wie aus den Anwendungsbeispiel Radio Frequency Identification - RFID (Kapitel 8.4) ersichtlich, ist diese Anforderung erfüllt.

Anforderung 8: Die Methode muss eine diskrete ereignisorientierte Simulationskomponente beinhalten, die in der Lage ist, sowohl Material- als auch Informationsflüsse der Supply Chain im Zeitverlauf zu simulieren.

Wie aus den Anwendungsbeispielen ersichtlich, ist diese Anforderung erfüllt.

Anforderung 9: Dynamische Veränderungen des Modells können durch ein mächtiges Regelwerk definiert werden, das während der Simulation ausgewertet wird und den Verlauf der Simulation beeinflussen kann.

Durch die Verwendung von LeoExpressions steht eine sehr umfangreiche Sprache zur Definition von Regeln zur Verfügung. Die Einbindung der Regel-Engine in die Simulation wird im konzeptionellen Modell theoretisch beschrieben und im Prototyp umgesetzt. Das Resultat ist in den Anwendungsbeispielen ersichtlich.

Anforderung 10: Die Animationskomponente muss dynamische Objekte wie Teile, Nachrichten und Abrufe direkt im Supply Chain Modell darstellen und deren Bewegung durch das Modell visualisieren können.

Wie aus den Anwendungsbeispielen ersichtlich, ist diese Anforderung erfüllt.

Anforderung 11: Die Animationskomponente sollte während der Simulation dem Nutzer Feedback über den Verlauf der Simulation geben und eine Playback-Animation nach der Simulation erlauben.

Sowohl das konzeptionelle Modell, als auch die implementierten Algorithmen erfüllen diese Anforderung.

Anforderung 12: Eine Beschreibung der Zustandsfolge eines Modells von einem Startzeitpunkt bis zu einem Endzeitpunkt sollte als Bericht generierbar sein.

Diese Funktionalität ist im konzeptionellen Modell beschrieben und im Prototyp so umgesetzt, dass die Zustandsfolge eines Modells innerhalb einer Periode dargestellt werden kann. Die Reihenfolge der Zustandsänderungen wird über sogenannte Microsteps gezeigt. Das Beispiel „Ersatzteileingang“, Kapitel 8.1, zeigt eine mögliche Darstellung des Ergebnisses dieser Funktionalität. Des Weiteren wird eine Funktionalität im Prototyp angeboten, mit der der Lebenszyklus eines dynamischen Teils des Materialflusses über mehrere Perioden hinweg gezeigt werden kann.

Diese Zusammenfassung der Ergebnisse zeigt, dass die gestellten Anforderungen mit der Methode SIMchronization erfüllt werden können. Demgemäß konnte auch die, in Kapitel 1.2, aufgestellte Forschungsfrage mit der Entwicklung und prototypischen Umsetzung der Methode positiv beantwortet werden.

9. Modellierungsverfahren

Zum Abschluss der Arbeit noch ein paar Hinweise zum Modellierungsverfahren. Gemäß dem, in Kapitel 2.1.1 beschriebenen, von Karagiannis und Kühn (Karagiannis und Kühn, 2002) vorgestelltem Generic Modelling Method Specification Framework, besteht eine Modellierungsmethode aus einer Modellierungssprache, Modellierungsprozeduren und Mechanismen und Algorithmen. Sowohl die Modellierungssprache als auch die Mechanismen und Algorithmen wurden in dieser Arbeit ausgiebig beschrieben, deren Ausgestaltung für die Methode SIMchronization hergeleitet und Alternativen diskutiert. Schlussendlich wurden mittels einer prototypischen Implementierung, die zuvor aufgestellten Anforderungen evaluiert.

Nur am Rande behandelt wurden die Modellierungsprozeduren, also Anweisungen, Richtlinien und Tipps zur Anwendung der Methode und zur korrekten Deutung der Resultate. In (Prackwieser, 2014) wurde ein erster, sehr grober Ansatz zur Anwendung der Methode vorgestellt:

Identifikation der Systemgrenzen: In Abhängigkeit vom Analyseziel muss zunächst der Scope der Untersuchung festgelegt werden. Dazu zählen die Start- und Endpunkte des Materialflusses in der Maintenance Supply Chain, der erforderliche Detailierungsgrad der Modellierung und die zu simulierende Zeitdauer.

Erstellung der statischen Modelle: Zunächst werden die, sich im Untersuchungsbereich befindlichen, statischen Objekte des Supply Chain Network Models modelliert. Wenn erforderlich kann parallel dazu ein Resource Model aufgebaut oder ein bestehendes wiederverwendet werden. Sofern zwischen statischen Objekten ein Materialfluss oder Informationsfluss zustande kommen könnte, werden die entsprechenden Relationen angelegt.

Annotation verbaler Beschreibungen: Die von Fachbereichsexperten oder sonstigen Stakeholder zur Verfügung gestellten zusätzlichen verbalen Informationen werden in die Beschreibungsattribute der statischen Objekte eingetragen und somit das graphische Modell durch textuelle Annotationen verfeinert. Besonders interessant ist hierbei Information bezüglich des Verhaltens der Objekte, deren Abhängigkeiten untereinander, verwendete Steuerungsverfahren und Kommunikation zwischen den Supply Chain Partnern.

Hinterlegung verhaltensbeschreibender Regeln und Programme: Die verbalen Beschreibungen werden herangezogen und wenn erforderlich, in die Regelsprache überführt.

Das Modell wird zur Simulation vorbereitet: Es erfolgt eine Prüfung, ob die Modelle syntaktisch korrekt sind und alle zur Simulation notwendigen Parameter angegeben wurden. Gegeben falls müssen zusätzliche Informationen gesammelt und in das Modell eingepflegt werden.

Simulationsausführung und Animation: Die Bewegung der Teile und ausgetauschte Informationen können nach der Simulation mittels der Animation beobachtet und analysiert werden. Diese Funktionalität bietet eine gute Unterstützung zur Transparentmachung der Dynamik und der Synchronisation von Material- und Informationsflüssen innerhalb des Modells. Fachexperten können das Verhalten des Systems mit ihren gesammelten Erfahrungen der ‚realen' Welt vergleichen und die Ergebnisse weiterverwenden.

Die oben aufgeführten Schritte zur Anwendung von SIMchronization sind als erster Schritt zur Entwicklung einer Modellierungsverfahrens zu sehen. Es besteht noch großer Forschungsbedarf in der zielgerichteten Anwendung der Methode und Ausgestaltung des Modellierungsverfahrens. Um dies und eine generelle Weiterentwicklung der Methode SIMchronization zu ermöglichen, wird die Methode SIMchronization mittels der Plattform Open Models Laboratory (OMiLAB) der wissenschaftlichen Community zur Verfügung gestellt. Damit verbunden ist die Hoffnung, dass diese Methode auf Interesse bei Studenten, Forschern und Praktikern stößt, sie Bestandteil zukünftige Forschungsprojekte wird und somit weiter verfeinert, entwickelt und auf andere Domänen angewendet wird.

10. Literaturverzeichnis

Ahuja, I., & Khamba, J. (2008). Total productive maintenance: literature review and directions. *International Journal of Quality & Reliability Management, 25*(7), 709-756.

Al-Safadi, L., & Al-Sulaiman, Z. (2011). RFID-based Clinical Decision Support System using Simulation Modeling. *Journal of Applied Sciences, 11*(15). doi:10.3923/jas.2011.2808.2815

Alabdulkarim, A. A., Ball, P. D., & Tiwari, A. (2011). *Rapid modeling of field maintenance using discrete event simulation.* Paper presented at the Simulation Conference (WSC), Proceedings of the 2011 Winter.

Almeder, C., & Preusser, M. (2007a). *A hybrid simulation optimization approach for supply chains.* Paper presented at the EUROSIM 2007, Ljubljana, Slovenia.

Almeder, C., & Preusser, M. (2007b). *A toolbox for simulation-based optimization of supply chains*. Paper presented at the Proceedings of the 39th conference on Winter simulation: 40 years! The best is yet to come, Washington D.C.

Almeder, C., Preusser, M., & Hartl, R. F. (2009). Simulation and optimization of supply chains: alternative or complementary approaches? *OR spectrum, 31*(1), 95-119.

Amini, M., Otondo, R. F., Janz, B. D., & Pitts, M. G. (2007). Simulation Modeling and Analysis: A Collateral Application and Exposition of RFID Technology. *Production and Operations Management, 16*(5), 586-598. doi:10.1111/j.1937-5956.2007.tb00282.x

Andijani, A., & Duffuaa, S. (2002). Critical evaluation of simulation studies in maintenance systems. *Production Planning & Control, 13*(4), 336-341. doi:10.1080/09537280110077578

APICS Supply Chain Council. (2015). *SCOR Supply Chain Operations Reference Model - Quick Reference Guide Version 11.0*. Retrieved from

Arnold, D., & Furmans, K. (2009). Abbildung von Materialflusssystemen in Modellen *Materialfluss in Logistiksystemen* (pp. 47-110). Berlin, Heidelberg: Springer Berlin Heidelberg.

Arnold, D., Furmans, K., Isermann, H., Kuhn, A., & Tempelmeier, H. (2008). *Handbuch Logistik* (3 ed.). Berlin, Heidelberg: Springer-Verlag.

Arns, M., Fischer, M., Kemper, P., & Tepper, C. (2002). Supply Chain Modelling and Its Analytical Evaluation. *The Journal of the Operational Research Society, 53*(8), 885-894.

Asif, Z. (2005). Integrating the supply chain with RFID: A technical and business analysis. *Communications of the Association for Information Systems, 15*(1), 24.

Atkinson, C., & Kühne, T. (2003). Model-driven development: a metamodeling foundation. *IEEE software, 20*(5), 36-41.

Baldwin, R. C. (2000). How Do You Spell e-Maintenenace? Retrieved from http://www.mt-online.com/april2000/how-do-you-spell-e-maintenenace

Banks, J. (1999). Introduction to simulation. *WSC'99. 1999 Winter Simulation Conference Proceedings. `Simulation - A Bridge to the Future' (Cat. No.99CH37038).* doi:10.1109/wsc.1999.823046

Banks, J., Buckley, S., Jain, S., Lendermann, P., & Manivannan, M. (2002, 8-11 Dec. 2002). *Panel session: opportunities for simulation in supply chain management.* Paper presented at the Simulation Conference, 2002. Proceedings of the Winter.

Banks, J., Carson II, J. S., Automation, B., Nelson, B. L., & Nicol, D. M. (2004). *Discrete-Event System Simulation, Fourth Edition*: Prentice Hall.

Banks, J., & Chwif, L. (2011). Warnings about simulation. *Journal of Simulation, 5*(4), 279-291. doi:10.1057/jos.2010.24

Banks, J., & Gibson, R. (1997). Don't Simulate When... 10 Rules for Determining when Simulation Is Not Appropriate-While simulation tools are appropriate for solving many types of problems, in some situations there are quicker. *IIE solutions, 29*(9), 30-33.

Barnett, W., Presley, A., Johnson, M., & Liles, D. (1994). *An architecture for the virtual enterprise.* Paper presented at the Systems, Man, and Cybernetics, 1994. Humans, Information and Technology., 1994 IEEE International Conference on.

Bause, F., Beilner, H., Fischer, M., Kemper, P., & Völker, M. (2002). The ProC/B Toolset for the Modelling and Analysis of Process Chains Computer Performance Evaluation: Modelling Techniques and Tools. In T. Field, P. Harrison, J. Bradley, & U. Harder (Eds.), (Vol. 2324, pp. 25-30): Springer Berlin / Heidelberg.

Becker, J., Rosemann, M., & Schütte, R. (1995). Grundsätze ordnungsmäßiger modellierung. *Wirtschaftsinformatik, 37*(5), 435-445.

Bergmann, S. (2014). *Automatische Generierung adaptiver Modelle zur Simulation von Produktionssystemen.* Ilmenau : Univ.-Verl. Ilmenau.

Bertin, J. (1983). Semiology of graphics: diagrams, networks, maps.

Bézivin, J., & Gerbé, O. (2001). *Towards a precise definition of the OMG/MDA framework.* Paper presented at the Automated Software Engineering, 2001.(ASE 2001). Proceedings. 16th Annual International Conference on.

Biethahn, J., Hummeltenberg, W., Schmidt, B., Stähly, P., & Witte, T. (2013). *Simulation als betriebliche Entscheidungshilfe: State of the Art und neuere Entwicklungen*: Springer-Verlag.

Bizer, C., Heath, T., & Berners-Lee, T. (2009). Linked data-the story so far. *Semantic services, interoperability and web applications: emerging concepts*, 205-227.

BOC Information Technologies Consulting AG. (2014). *ADONIS(R) Version 6.0 Band IV: Methodendefinition und Administration*

Bornhövd, C., Lin, T., Haller, S., & Schaper, J. (2004). *Integrating automatic data acquisition with business processes experiences with SAP's auto-ID infrastructure.* Paper presented at the Proceedings of the Thirtieth international conference on Very large data bases-Volume 30.

Bosch, D. (2016). Der Instandhalter – mehr als nur kreativer Problemlöser im Dickicht der Begriffe. *Instandhaltung*, 8. Retrieved from www.bosch-me.de/papers/fachartikel-der-instandhalter.pdf

Bossel, H. (2004). *Systeme, Dynamik, Simulation: Modellbildung, Analyse und Simulation komplexer Systeme*: BoD–Books on Demand.

Breitling, H., Kornstadt, A., & Sauer, J. (2006). Design Rationale in Exemplary Business Process Modeling. *Rationale management in software engineering*, 191.

Brinkkemper, S. (1996). Method engineering: engineering of information systems development methods and tools. *Information and Software Technology, 38*(4), 275-280.

Brooks, R. J., & Tobias, A. M. (1996). Choosing the best model: Level of detail, complexity, and model performance. *Math. Comput. Model., 24*(4), 1-14. doi:10.1016/0895-7177(96)00103-3

Bruckner, A., & Spille, J. (2004). LogiBest—Referenzmodell logistischer Prozesse für branchenübergreifendes Benchmarking *Betriebsorganisation im Unternehmen der Zukunft* (pp. 305-318): Springer.

Buchmann, R. A. (2014). *Conceptual modeling for mobile maintenance: the ComVantage case.* Paper presented at the 2014 47th Hawaii International Conference on System Sciences.

Buchmann, R. A., & Karagiannis, D. (2015). *Agile Modelling Method Engineering: Lessons Learned in the ComVantage Research Project.* Paper presented at the IFIP Working Conference on The Practice of Enterprise Modeling.

Budinsky, F., Steinberg, D., Merks, E., Ellersick, R., & Grose, T. (2003). *Eclipse Modeling Framework*: Addison Wesley Professional.

Carson, J. S. (2002). *Model verification and validation.* Paper presented at the Simulation Conference, 2002. Proceedings of the Winter.

Casti, J. L. (1979). *Connectivity, complexity and catastrophe in large-scale systems*: John Wiley & Sons.

Christopher, M. (2005). *Logistics and supply chain management: creating value-added networks*: Pearson education.

Christopher, M. (2016). *Logistics & supply chain management*: Pearson Higher Ed.

Cook, S., Jones, G., Kent, S., & Wills, A. C. (2007). *Domain-specific development with visual studio dsl tools*: Pearson Education.

Cooper, M. C., Lambert, D. M., & Pagh, J. D. (1997). Supply chain management: more than a new name for logistics. *The International Journal of Logistics Management, 8*(1), 1-14.

Davis, A. M. (1988). A comparison of techniques for the specification of external system behavior. *Commun. ACM, 31*(9), 1098-1115. doi:10.1145/48529.48534

Dekker, R. (1996). Applications of maintenance optimization models: a review and analysis. *Reliability Engineering & System Safety, 51*(3), 229-240. doi:10.1016/0951-8320(95)00076-3

Dijkman, R. M., Dumas, M., & Ouyang, C. (2008). Semantics and analysis of business process models in BPMN. *Information and Software Technology, 50*(12), 1281-1294. doi:10.1016/j.infsof.2008.02.006

DIN Deutsches Institut für Normung e. V. (2010). DIN EN 13306: Instandhaltung – Begriffe der Instandhaltung; Dreisprachige Fassung EN 13306:2010 (Deutsche Norm) (Vol. DIN EN 13306): DIN Deutsches Institut für Normung e. V.,.

DIN Deutsches Institut für Normung e. V. (2011). Grundlagen der Instandhaltung (Vol. DIN 31051 / Entwurf Einspruchsfrist bis 2012-04-05, pp. 12): DIN Deutsches Institut für Normung e. V.

Duffuaa, S., Ben-Daya, M., Al-Sultan, K., & Andijani, A. (2001). A generic conceptual simulation model for maintenance systems. *Journal of Quality in Maintenance Engineering, 7*(3), 207-219.

Efendioglu, N., Woitsch, R., & Karagiannis, D. (2015). *Modelling method design: a model-driven approach*. Paper presented at the Proceedings of the 17th International Conference on Information Integration and Web-based Applications & Services, Brussels, Belgium.

Ehrig, K. (2003). *Konzeption und Implementierung eines Generators für Animationsumgebungen für visuelle Modellierungssprachen*: Techn. Univ. Berlin, Fakultät IV, Elektrotechnik und Informatik.

Erlach, K. (2010). *Wertstromdesign : der Weg zur schlanken Fabrik* (2., bearb. und erw. Aufl. ed.). Berlin [u.a.]: Springer.

Espíndola, D., Frazzon, E. M., Hellingrath, B., & Pereira, C. E. (2012). Integrating intelligent maintenance systems and spare parts supply chains. *IFAC Proceedings Volumes, 45*(6), 1017-1022.

Fabry, C., & Schmitz-Urban, A. (2010). *Maintenance Supply Chain Optimisation within an IT-Platform: Network Service Science and Management*. Paper presented at the 2010 International Conference on Management and Service Science (MASS).

Fadel, F. G., Fox, M. S., & Gruninger, M. (1994, 17-19 Apr 1994). *A generic enterprise resource ontology.* Paper presented at the Enabling Technologies: Infrastructure for Collaborative Enterprises, 1994. Proceedings., Third Workshop on.

Fettke, P. (2007). Supply Chain Management: Stand der empirischen Forschung. *Zeitschrift für Betriebswirtschaft, 77*(4), 417-461.

Fill, H.-G. (2009). *Visualisation for Semantic Information Systems*: Gabriel Gurley.

Fill, H.-G., Hickl, S., Karagiannis, D., Oberweis, A., & Schoknecht, A. (2013a). *A Formal Specification of the Horus Modeling Language Using FDMM.* Paper presented at the Wirtschaftsinformatik.

Fill, H.-G., & Karagiannis, D. (2013). On the conceptualisation of modelling methods using the ADOxx meta modelling platform. *Enterprise Modelling and Information Systems Architectures-An International Journal, 8*(1).

Fill, H. G., Redmond, T., & Karagiannis, D. (2012). *Formalizing Meta Models with FDMM: The ADOxx Case.* Paper presented at the International Conference on Enterprise Information Systems.

Fill, H. G., Redmond, T., & Karagiannis, D. (2013b). Formalizing Meta Models with FDMM: The ADOxx Case. In J. Cordeiro, L. A. Maciaszek, & J. Filipe (Eds.), *Enterprise Information Systems, Iceis 2012* (Vol. 141, pp. 429-451). Berlin: Springer-Verlag Berlin.

Finkenzeller, K. (2015). *RFID-Handbuch: Grundlagen und praktische Anwendungen von Transpondern, kontaktlosen Chipkarten und NFC*: Carl Hanser Verlag GmbH Co KG.

Flood, R. L., & Carson, E. (2013). *Dealing with complexity: an introduction to the theory and application of systems science*: Springer Science & Business Media.

Ford, M., & GmbH, U. C. S. (1998). *Handbuch Netzwerk-Technologien:[komplettes Grundwissen für Internetworking und Networking]*: Markt-und-Technik, Buch-und Software-Verlag.

Forrester, J. W. (1969). *Industrial dynamics* (6. print. ed.). Cambridge, Mass.: MIT Press.

Fowler, M. (2004). *UML distilled: a brief guide to the standard object modeling language*: Addison-Wesley Professional.

Fox, M. S., Barbuceanu, M., & Gruninger, M. (1996). An organisation ontology for enterprise modeling: Preliminary concepts for linking structure and behaviour. *Computers in Industry, 29*(1–2), 123-134. doi:10.1016/0166-3615(95)00079-8

Frank, U. (2013). Domain-specific modeling languages: requirements analysis and design guidelines *Domain Engineering* (pp. 133-157): Springer.

Frank, U., & Bodo, v. L. (2003). *Anforderungen an Sprachen zur Modellierung von Geschäftsprozessen*. Retrieved from Arbeitsberichte des Instituts für Wirtschaftsinformatik, Universität Koblenz:

Golay, M. W., Seong, P. H., & Manno, V. P. (1989). A measure of the difficulty of system diagnosis and its relationship to complexity. *International Journal Of General System, 16*(1), 1-23.

Götzinger, D., Miron, E.-T., & Staffel, F. (2016). OMiLAB: An Open Collaborative Environment for Modeling Method Engineering *Domain-Specific Conceptual Modeling* (pp. 55-76): Springer.

Greeff, G., & Ghoshal, R. (2004). *Practical e-manufacturing and supply chain management Practical professional books from Elsevier* Retrieved from http://www.sciencedirect.com/science/book/9780750662727

Grubic, T., & Fan, I.-S. (2010). Supply chain ontology: Review, analysis and synthesis. *Computers in Industry, 61*(8), 776-786. doi:10.1016/j.compind.2010.05.006

Günthner, W., & Schneider, O. (2011). Methode zur einfachen Aufnahme und intuitiven Visualisierung innerbetrieblicher logistischer Prozesse. *Forschungsbericht, Lehrstuhl für Fördertechnik Materialfluss Logistik, TU München*.

Gutzwiller, T. A. (1994). *Das CC-RIM Referenzmodell für den Entwurf von betrieblichen, transaktionsorientierten Informationssystemen*. Heidelberg: Physica.

Harnisch, M. J., & Uitz, I. (2013). Near Field Communication (NFC). *Informatik-Spektrum, 36*(1), 99-103. doi:10.1007/s00287-012-0672-x

Hausladen, I., & Bechheim, C. (2004, 26-26 June 2004). *E-maintenance platform as a basis for business process integration.* Paper presented at the Industrial Informatics, 2004. INDIN '04. 2004 2nd IEEE International Conference on.

Hawryszkiewycz, I. T., & Prackwieser, C. (2016). MELCA - Customizing Visualizations for Design Thinking *Domain-Specific Conceptual Modeling* (pp. 383-396): Springer.

Hay, D., & Healy, K. A. (2000). *Defining Business Rules - What Are They Really?* Retrieved from

Hedtstück, U. (2013). Simulationstechniken für diskrete Prozesse *Simulation diskreter Prozesse: Methoden und Anwendungen* (pp. 21-37). Berlin, Heidelberg: Springer Berlin Heidelberg.

Heiserich, O.-E., Helbig, K., & Ullmann, W. (2011). Logistik. Eine praxisorientierte Einführung. *4., vollständig überarbeitete und erweiterte Auflage*: Wiesbaden: Gabler Verlag/Springer Fachmedien Wiesbaden GmbH Wiesbaden. S.

Hellingrath, B., & Cordes, A.-K. (2014). Conceptual approach for integrating condition monitoring information and spare parts forecasting methods. *Production & Manufacturing Research, 2*(1), 725-737.

Henderson-Sellers, B., & Ralyté, J. (2010). Situational method engineering: state-of-the-art review. *Journal of Universal Computer Science, 16*(3), 424-478.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Q., 28*(1), 75-105.

Heym, M. (1993). *Methoden-Engineering: Spezifikation und Integration von Entwicklungsmethoden für Informationssysteme.* (Dissertation), Universität St. Gallen, Hallstadt

Hinkelmann, K., Albayrak, M., Kritikos, K., Sabrina, K., Lammel, B., & Woitsch, R. (2015). *MODELLING FRAMEWORK FOR BPAAS D3.1*. Retrieved from

Hinz, S., Schmidt, K., & Stahl, C. (2005). Transforming BPEL to Petri Nets. In W. P. Aalst, B. Benatallah, F. Casati, & F. Curbera (Eds.), *Business Process Management* (Vol. 3649, pp. 220-235): Springer Berlin Heidelberg.

Hofmann, E., Beck, P., & Füger, E. (2012). *The supply chain differentiation guide: a roadmap to operational excellence*: Springer Science & Business Media.

Holten, R. (2001). *Konstruktion domänenspezifischer Modellierungstechniken für die Modellierung von Fachkonzepten*. Retrieved from Münster:

Huang, G. Q., Lau, J. S. K., & Mak, K. L. (2003). The impacts of sharing production information on supply chain dynamics: A review of the literature. *International Journal of Production Research, 41*(7), 1483-1517. doi:10.1080/0020754031000069625

Huiskonen, J. (2001). Maintenance spare parts logistics: Special characteristics and strategic choices. *International Journal of Production Economics, 71*(1–3), 125-133. doi:10.1016/s0925-5273(00)00112-2

Ingalls, R. G. (1998). *The value of simulation in modeling supply chains.* Paper presented at the Proceedings of the 30th conference on Winter simulation.

Jacobs, F. R. (2000). Playing the beer distribution game over the internet. *Production and Operations Management, 9*(1), 31.

Jain, S., Workman, R. W., Collins, L. M., Ervin, E. C., & Lathrop, A. P. (2001). *Supply chain applications II: development of a high-level supply chain simulation model*. Paper presented at the Proceedings of the 33nd conference on Winter simulation, Arlington, Virginia.

Jakkhupan, W., Arch-int, S., & Li, Y. (2011). Business process analysis and simulation for the RFID and EPCglobal Network enabled supply chain: A proof-of-concept approach. *Journal of Network and Computer Applications, 34*(3), 949-957. doi:10.1016/j.jnca.2010.04.003

Jakoby, W. (2013). *Automatisierungstechnik—Algorithmen und Programme: Entwurf und Programmierung von Automatisierungssystemen*: Springer-Verlag.

Jendoubi, L. (2007). *Management mobiler Betriebsmittel unter Einsatz ubiquitärer Computersysteme in der Produktion.* Universitätsbibliothek der Universität Stuttgart, Stuttgart. Retrieved from http://elib.uni-stuttgart.de/opus/volltexte/2007/3195

Jin, D., Hongwei, D., Changrui, R., & Wei, W. (2006). IBM SmartSCOR - a SCOR based supply chain transformation platform through simulation and optimization techniques. *Proceedings of the 2006 Winter Simulation Conference (IEEE Cat No. 06CH37826)*.

Joshi, Y. V. (2000). *Information visibility and its effect on supply chain dynamics.* Massachusetts Institute of Technology.

Junginger, S., Kühn, H., Strobl, R., & Karagiannis, D. (2000). Ein Geschäftsprozessmanagement-Werkzeug der nächsten Generation - ADONIS: Konzeption und Anwendungen. *Wirtschaftsinformatik, 42*(5), 392-401.

Junginger, S. G. (2001). *Workflowbasierte Umsetzung von Geschäftsprozessen*: na.

Kaczmarek, M. (2002). *Definition von Anforderungen an die Modellierung und Analyse der Supply Chain Technical Report 02007*

Karagiannis, D. (2015). *Agile modeling method engineering.* Paper presented at the Proceedings of the 19th Panhellenic Conference on Informatics.

Karagiannis, D., Buchmann, R. A., Burzynski, P., Reimer, U., & Walch, M. (2016a). Fundamental Conceptual Modeling Languages in OMiLAB *Domain-Specific Conceptual Modeling* (pp. 3-30): Springer.

Karagiannis, D., Grossmann, W., & Höfferer, P. (2007). *Open Model Initiative - A Feasibility Study, Project Study on behalf of the Austrian Federal Ministry for Transport, Innovation and Technology*. Retrieved from http://cms.dke.univie.ac.at/uploads/media/Open_Models_Feasibility_Study_SEPT_2008.pdf (Zugriff 08.01.2017):

Karagiannis, D., & Höfferer, P. (2006, September 11 – 14, 2006). *Metamodels in action: An overview.* Paper presented at the International Conference on Software and Data Technologies, Setúbal, Portugal.

Karagiannis, D., & Kühn, H. (2002). Metamodelling platforms. In K. Bauknecht, A. M. Tjoa, & G. Quirchmayr (Eds.), *E-Commerce and Web Technologies, Proceedings* (Vol. 2455, pp. 182-182).

Karagiannis, D., Mayr, H. C., & Mylopoulos, J. (2016b). Domain-Specific Conceptual Modelling: Springer.

Karagiannis, D., Prackwieser, C., & Telesko, R. (2000). *The PROMOTE project: Process oriented knowledge management.* Paper presented at the Proceedings of the 3rd European Conference on Product and Process Modeling.

Karagiannis, D., & Visic, N. (2011). Next Generation of Modelling Platforms. Perspectives in Business Informatics Research. In J. Grabis & M. Kirikova (Eds.), (Vol. 90, pp. 19-28): Springer Berlin Heidelberg.

Karim, R., Candell, O., & Söderholm, P. (2009). E-maintenance and information logistics: aspects of content format. *Journal of Quality in Maintenance Engineering, 15*(3), 308-324.

Kelly, S., Rossi, M., & Tolvanen, J.-P. (2005). What is Needed in a MetaCASE Environment? *Enterprise Modelling and Information Systems Architectures, 1*(1), 25-35.

Kennedy, W. J., Wayne Patterson, J., & Fredendall, L. D. (2002). An overview of recent literature on spare parts inventories. *International Journal of Production Economics, 76*(2), 201-215. doi:10.1016/s0925-5273(01)00174-8

Kern, H., Hummel, A., & Kühne, S. (2011). *Towards a comparative analysis of meta-metamodels.* Paper presented at the Proceedings of the compilation of the co-located workshops on DSM'11, TMC'11, AGERE! 2011, AOOPES'11, NEAT'11, & VMIL'11.

Kim, K.-Y., Manley, D. G., & Yang, H. (2006). Ontology-based assembly design and information sharing for collaborative product development. *Computer-Aided Design, 38*(12), 1233-1250. doi:10.1016/j.cad.2006.08.004

Kirchner, L. (2007). *Eine Methode zur Unterstützung des IT-Managements im Rahmen der Unternehmensmodellierung.* (PhD Thesis), University of Duisburg-Essen.

Kleijnen, J. P. C. (2005). Supply chain simulation tools and techniques: a survey. *International Journal of Simulation and Process Modelling, 1*(1), 82-89.

Kleijnen, J. P. C., & Smits, M. T. (2003). Performance metrics in supply chain management. *Journal of the Operational Research Society, 54*(5), 507-514. doi:DOI 10.1057/palgrave.jors.2601539

Klevers, T. (2007). *Wertstrom-Mapping und Wertstrom-Design : Verschwendung erkennen - Wertschöpfung steigern*. Landsberg am Lech: mi-Fachverl.

Klimek, G. (2011). *Maintenance Supply Chain Optimisation - MSCO : Abschlussbericht ; Entwicklung eines Logistikkonzeptes zur Optimierung des Ersatzteilmanagement in der Instandhaltung durch Integration aller am Geschäftprozess Beteiligten und die Synchronisation der gesamten Lieferkette ; Laufzeit des Vorhabens: 01.08.2007 - 31.10.2010 / Projektkonsortium: Forschungsinstitut für Rationalisierung (FIR) e.V. an der RWTH Aachen (Konsortialführer) ...* Retrieved from Aachen: http://publications.rwth-aachen.de/record/206515

Koç, M., & Ni, J. (2005). *Introduction to e-manufacturing*.

Košturiak, J., & Gregor, M. (1995). Simulation von Produktionssystemen *Simulation von Produktionssystemen* (pp. 71-103). Vienna: Springer Vienna.

Krafcik, J. F. (1988). Triumph of the lean production system. *MIT Sloan Management Review, 30*(1), 41.

Kruse, C., & Scheer, A.-W. (1992). *Modellierung und Analyse dynamischen Systemverhaltens*: Inst. für Wirtschaftsinformatik.

Kuhn, A., & Hellingrath, B. (2002). *Supply Chain Management: optimierte Zusammenarbeit in der Wertschöpfungskette*. Berlin [u.a.]: Springer.

Kuhn, A., Schuh, G., & Stahl, B. (2006). *Nachhaltige Instandhaltung: Trends, Potenziale und Handlungsfelder nachhaltiger Instandhaltung ; [Ergebnisbericht der vom BMBF geförderten Untersuchung "Nachhaltige Instandhaltung"]*. Frankfurt a.M.: VDMA-Verl.

Kurpjuweit, S., & Winter, R. (2007). *Viewpoint-based Meta Model Engineering.* Paper presented at the EMISA.

Lambert, D. M., Cooper, M. C., & Pagh, J. D. (1998). Supply Chain Management: Implementation Issues and Research Opportunities. *The International Journal of Logistics Management, 9*(2), 1-20. doi:doi:10.1108/09574099810805807

Landt, J. (2005). The history of RFID. *IEEE Potentials, 24*(4), 8-11. doi:10.1109/MP.2005.1549751

Law, A. M. (2008). *How to build valid and credible simulation models.* Paper presented at the Proceedings of the 40th Conference on Winter Simulation.

Law, A. M., & Kelton, W. D. (1991). *Simulation modeling and analysis* (Vol. 2): McGraw-Hill New York.

Lee, H. L., Padmanabhan, V., & Whang, S. (1997). Information Distortion in a Supply Chain: The Bullwhip Effect. *Management Science, 43*(4), 546-558.

Lee, J. (2001, 2001). *A framework for Web-enabled e-maintenance systems.* Paper presented at the Environmentally Conscious Design and Inverse Manufacturing, 2001. Proceedings EcoDesign 2001: Second International Symposium on.

Lee, J., Ni, J., Djurdjanovic, D., Qiu, H., & Liao, H. (2006). Intelligent prognostics tools and e-maintenance. *Computers in Industry, 57*(6), 476-489.

Lee, Y. M., Cheng, F., & Leung, Y. T. (2005). A quantitative view on how RFID will improve a supply chain. *IBM Research Report*, 1-45.

Lehner, F., Hildebrand, K., & Maier, R. (1995). *Wirtschaftsinformatik: Theoretische Grundlagen*: Hanser.

Leung, Y. T., Cheng, F., Lee, Y. M., & Hennessy, J. J. (2007). A tool set for exploring the value of RFID in a supply chain *Trends in supply chain design and management* (pp. 49-70): Springer.

Leutgeb, A., Utz, W., Woitsch, R., & Fill, H.-G. (2007). ADAPTIVE PROCESSES IN E-GOVERNMENT.

Levrat, E., Iung, B., & Crespo Marquez, A. (2008). E-maintenance: review and conceptual framework. *Production Planning & Control, 19*(4), 408-429. doi:10.1080/09537280802062571

Li, X., & Wang, Q. (2007). Coordination mechanisms of supply chain systems. *European Journal of Operational Research, 179*(1), 1-16.

Liebl, F. (1992). *Simulation : problemorientierte Einführung*. München [u.a.]: Oldenbourg.

Lin, Y.-C., Cheung, W.-F., & Siao, F.-C. (2014). Developing mobile 2D barcode/RFID-based maintenance management system. *Automation in Construction, 37*, 110-121. doi:http://dx.doi.org/10.1016/j.autcon.2013.10.004

Lindemann, M., Junginger, S., Rausch, T., & Kühn, H. (2002). *ADOlog: An Implementation of the Supply Chain Operations Reference Model.* Paper presented at the Proceedings of the 9th European Concurrent Engineering Conference.

Lohmann, N., Verbeek, E., & Dijkman, R. (2009). Petri net transformations for business processes–a survey *Transactions on petri nets and other models of concurrency II* (pp. 46-63): Springer.

Madenas, N., Tiwari, A., Turner, C. J., & Woodward, J. (2014). Information flow in supply chain management: A review across the product lifecycle. *CIRP Journal of Manufacturing Science and Technology, 7*(4), 335-346. doi:http://dx.doi.org/10.1016/j.cirpj.2014.07.002

Madni, A. M., Lin, W., & Madni, C. C. (2001). IDEONTM: An extensible ontology for designing, integrating, and managing collaborative distributed enterprises. *Systems Engineering, 4*(1), 35-48. doi:10.1002/1520-6858(2001)4:1<35::aid-sys4>3.0.co;2-f

Mebes, P. (2008). *Graphen-basierte Modellierungsmethode zur Materialflussverfolgung.*

Meier, C., & Klimek, G. (2008). MSCO: Maintenance Supply Chain Optimisation. *UdZ*, 24.

Meixell, M. J., Shaw, N. C., & Tuggle, F. D. (2008). A methodology for assessing the value of knowledge in a service parts supply chain. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 38*(3), 446-460.

Mentzer, J. T., DeWitt, W., Keebler, J. S., Min, S., Nix, N. W., Smith, C. D., & Zacharia, Z. G. (2001). Defining supply chain management. *Journal of Business logistics, 22*(2), 1-25.

Meyer, U. B., Creux, S. E. M., & Weber Marin Silva, A. K. (2005). *Grafische Methoden der Prozessanalyse : für Design und Optimierung von Produktionssystemen*. München [u.a.]: Hanser.

Molenaers, A., Baets, H., Pintelon, L., & Waeyenbergh, G. (2011). Criticality classification of spare parts: A case study. *International Journal of Production Economics*(0). doi:10.1016/j.ijpe.2011.08.013

Moody, D. (2009). The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering. *IEEE Transactions on Software Engineering, 35*(6), 756-779.

Muckstadt, J. A. (2005). *Analysis and algorithms for service parts supply chains*. New York, NY: Springer.

Muller, A., Crespo Marquez, A., & Iung, B. (2008). On the concept of e-maintenance: Review and current research. *Reliability Engineering & System Safety, 93*(8), 1165-1187. doi:10.1016/j.ress.2007.08.006

Müller, R. (1983). Zur Geschichte des Modelldenkens und des Modellbegriffs. *Modelle–Konstruktion der Wirklichkeit. München: Fink*, 17-86.

Münch, T., Hladik, J., Salmen, A., Altmann, W., Buchmann, R., Karagiannis, D., . . . Lazaro, O. (2014). Collaboration and interoperability within a virtual enterprise applied in a mobile maintenance scenario. *Revolutionizing Enterprise Interoperability through Scientific Foundations, 7*.

Nakajima, S. (1988). Introduction to TPM: Total Productive Maintenance.(Translation). *Productivity Press, Inc., 1988*, 129.

Naslund, D., & Williamson, S. (2010). What is management in supply chain management?-a critical review of definitions, frameworks and terminology. *Journal of Management Policy and Practice, 11*(4), 11-28.

Noy, N. F., & McGuinness, D. L. (2001). Ontology development 101: A guide to creating your first ontology. Retrieved from http://protege.stanford.edu/publications/ontology_development/ontology101.pdf

Object Management Group (OMG). (2014). Model Driven Architecture (MDA) *MDA Guide rev. 2.0* (Vol. OMG Document ormsc/2014-06-01).

Ohbuchi, E., Hanaizumi, H., & Hock, L. A. (2004, 18-20 Nov. 2004). *Barcode readers using the camera device in mobile phones.* Paper presented at the 2004 International Conference on Cyberworlds.

OMG. (2008). Semantics of Business Vocabulary and Business Rules (SBVR) *Version 1.0* (Vol. formal/2008-01-02).

OMG. (2009). Production Rule Representation (PRR) *Version 1.0* (Vol. formal/2009-12-01).

OMG. (2015). OMG Unified Modeling Language TM (OMG UML) *Version 2.5*.

Open Models Laboratory. (2014). *Open Models Laboratory Booklet Modelling Methods - ADOxx Platform* Retrieved from http://www.openmodels.at/web/guest/booklet

Owen, C. (2013). *Selection of simulation tools for improving supply chain performance.* (PhD), Aston University.

Page, B. (2013). *Diskrete Simulation: Eine Einführung mit Modula-2*: Springer-Verlag.

Park, O.-c., & Gittelman, S. (1992). Selective use of animation and feedback in computer-based instruction. *Educational Technology Research and Development, 40*(4), 27-38. doi:10.1007/bf02296897

Parlings, M., Hegmanns, T., Sprenger, P., & Kossmann, D. (2016). Modular IT-Support for Integrated Supply Chain Design *Logistics and Supply Chain Innovation* (pp. 295-312): Springer.

Parlings, M., & Motta, M. (2015). *Domänenspezifische Sprache für ein simulationsunterstütztes Supply Chain Design*. Paper presented at the Simulation in Production and Logistics 2015, Dortmund.

Paul Brunet, A., & New, S. (2003). Kaizen in Japan: an empirical study. *International Journal of Operations & Production Management, 23*(12), 1426-1446. doi:10.1108/01443570310506704

Pérès, F., & Noyes, D. (2006). Envisioning e-logistics developments: Making spare parts in situ and on demand: State of the art and guidelines for future developments. *Computers in Industry, 57*(6), 490-503. doi:10.1016/j.compind.2006.02.010

Persson, F. (2011). SCOR template—A simulation based dynamic supply chain analysis tool. *International Journal of Production Economics, 131*(1), 288-294. doi:10.1016/j.ijpe.2010.09.029

Persson, F., & Araldi, M. (2009). The development of a dynamic supply chain analysis tool—Integration of SCOR and discrete event simulation. *International Journal of Production Economics, 121*(2), 574-583. doi:10.1016/j.ijpe.2006.12.064

Persson, F., Bartoll, C., Ganovic, A., Nilsson, M., Wibaeus, J., & Winge, F. (2012). *Supply chain dynamics in the SCOR model—A simulation modeling approach.* Paper presented at the Simulation Conference (WSC), Proceedings of the 2012 Winter.

Petsch, M., Pawlaszczyk, D., & Schorcht, H. (2007). Regelbasierte Koordinierung von agentengestützten Transportprozessen. *Wirtschaftinformatik Proceedings 2007*, 77.

Ponis, S. T., Gayialis, S., Tatsiopoulos, I., Panayiotou, N., & Stamatiou, D. (2013). *Modeling supply chain processes: a review and critical evaluation of available reference models.* Paper presented at the 2nd International Symposium and 24th National Conference on Operational Research.

Prackwieser, C. (2013). *SIMchronization: Eine Modellierungsmethode zur Synchronisation von Material- und Informationsflüssen in der Instandhaltung*. Paper presented at the 11th International Conference on Wirtschaftsinformatik, Leipzig.

Prackwieser, C. (2014). A Modeling Procedure for Information and Material Flow Analysis Comprising Graphical Models, Rules and Animated Simulation. In R. Buchmann, C. V. Kifor, & J. Yu (Eds.), *Knowledge Science, Engineering and Management, Ksem 2014* (Vol. 8793, pp. 198-209). Berlin: Springer-Verlag Berlin.

Prackwieser, C., Buchmann, R., Grossmann, W., & Karagiannis, D. (2014). Overcoming Heterogeneity in Business Process Modeling with Rule-Based Semantic Mappings. *International Journal of Software Engineering and Knowledge Engineering, 24*(8), 1131-1158. doi:10.1142/s0218194014400087

Prackwieser, C., Buchmann, R. A., Grossmann, W., & Karagiannis, D. (2013). *Towards a Generic Hybrid Simulation Algorithm Based on a Semantic Mapping and Rule Evaluation Approach*. Paper presented at the The 2013 International Conference on Knowledge Science, Engineering and Management, Dalian, China. http://eprints.cs.univie.ac.at/3909/

Rabe, M., Spieckermann, S., & Wenzel, S. (2008). *Verifikation und Validierung für die Simulation in Produktion und Logistik: Vorgehensmodelle und Techniken*: Springer.

Rasch, A. A. (2000). *Erfolgspotential Instandhaltung: theoretische Untersuchung und Entwurf eines ganzheitlichen Instandhaltungsmanagements*. Berlin: Erich Schmidt.

Robinson, S. (2014). *Simulation: the practice of model development and use*: Palgrave Macmillan.

Robinson, S., & Higton, N. (1995). Computer simulation for quality and reliability engineering. *Quality and Reliability Engineering International, 11*(5), 371-377. doi:10.1002/qre.4680110508

Rohrer, M. W. (2000, 2000). *Seeing is believing: the importance of visualization in manufacturing simulation.* Paper presented at the Simulation Conference, 2000. Proceedings. Winter.

Rosenberg, F., & Dustdar, S. (2005). *Business Rules Integration in BPEL - A Service-Oriented Approach*. Paper presented at the Proceedings of the Seventh IEEE International Conference on E-Commerce Technology.

Rother, M., & Shook, J. (2003). *Learning to see: value stream mapping to add value and eliminate muda*: Lean Enterprise Institute.

Rother, M., & Shook, J. (2015). *Sehen lernen: mit Wertstromdesign die Wertschöpfung erhöhen und Verschwendung beseitigen* (Dt. Ausg., Vers. 1.1 ed.). Aachen: Lean Management Institut.

Saalmann, P., Wagner, C., & Hellingrath, B. (2016). Decision Support for a Spare Parts Supply Chain Coordination Problem: Designing a Tactical Collaborative Planning Concept. *IFAC-PapersOnLine, 49*(12), 1056-1061. doi:http://dx.doi.org/10.1016/j.ifacol.2016.07.582

Sahin, F., & Robinson, E. P. (2002). Flow Coordination and Information Sharing in Supply Chains: Review, Implications, and Directions for Future Research. *Decision Sciences, 33*(4), 505-536. doi:10.1111/j.1540-5915.2002.tb01654.x

Sarac, A., Absi, N., & Dauzere-Peres, S. (2015). Impacts of RFID technologies on supply chains: a simulation study of a three-level supply chain subject to shrinkage and delivery errors. *European Journal of Industrial Engineering, 9*(1), 27-52.

Schacher, M., & Grässle, P. (2006). *Agile Unternehmen durch Business Rules : Der Business Rules Ansatz Xpert.press* Retrieved from http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10183043

http://dx.doi.org/10.1007/3-540-32505-0

Schlichting, A. (2004). *Produktivitätsmanagement im Baubetrieb durch angewandte Kybernetik: Eine Organisationstheorie mit Anwendungsmöglichkeiten auch im praktischen Landschaftsbau*: diplom. de.

Schneider, H. M., Buzacott, J. A., Rücker, T., & Schneider Buzacott, R. (2005). *Operative Produktionsplanung und -steuerung : Konzepte und Modelle des Informations- und Materialflusses in komplexen Fertigungssystemen*. München [u.a.]: Oldenbourg.

Schneider O, H. F., Günthner WA. (2011). Bewertung von Methoden hinsichtlich einer ganzheitlichen Prozessdarstellung. . *Logistics Journal: Not reviewed publication, 2011*.

Schönherr, O., & Rose, O. (2010). Modellierung mit SysML zur Abbildung von Produktionsprozessen. *Integrationsaspekte der Simulation: Technik, Organisation und Personal, Gert Zülch & Patricia Stock (Hrsg.) Karlsruhe: KIT Scientific Publishing*, 453-460.

Schuh, G. (2006). Sm@rt Logistics: Intelligent networked systems. *CIRP Annals - Manufacturing Technology, 55*(1), 505-508. doi:http://dx.doi.org/10.1016/S0007-8506(07)60469-7

Schuh, G., & Attig, P. (2009). *Sm@rt Logistics : Synchronisation von Material- und Informationsfluss in der Produktion ; [Forschungsbericht]*. Aachen: Apprimus-Verl.

Schuh, G., Kampker, A., Franzkoch, B., & Wemhöner, N. (2005). Intelligent Maintenance–Potenziale zustandsorientierter Instandhaltung. *Abschlussbericht. Aachen*.

Schunk, D., & Plott, B. (2000). *Using simulation to analyze supply chains.* Paper presented at the Simulation Conference, 2000. Proceedings. Winter.

Schütte, R. (1999). *Zum Realitätsbezug von Informationsmodellen.* Paper presented at the EMISA Forum.

Schwermer, M. (1998). *Modellierungsvorgehen zur Planung von Geschäftsprozessen*: Fraunhofer Institut Produktionsanlagen und Konstruktionstechnik, IPK Berlin.

Schwetman, H. (1986). *CSIM: a C-based process-oriented simulation language.* Paper presented at the Proceedings of the 18th conference on Winter simulation.

Seidewitz, E. (2003). What models mean. *IEEE software, 20*(5), 26.

Shah, R., & Ward, P. T. (2007). Defining and developing measures of lean production. *Journal of Operations Management, 25*(4), 785-805.

Simon, H. A. (1991). The architecture of complexity *Facets of systems science* (pp. 457-476): Springer.

Sommer, L. (2011). The theory of planned behaviour and the impact of past behaviour. *The International Business & Economics Research Journal*, *10*(1), 91.

Soon, T. J. (2008). QR code. *Synthesis Journal*, *2008*, 59-78.

Stachowiak, H. (1974). *Allgemeine Modelltheorie*. Wien/New York: Springer.

Stadtler, H. (2005). Supply chain management and advanced planning—basics, overview and challenges. *European Journal of Operational Research, 163*(3), 575-588. doi:http://dx.doi.org/10.1016/j.ejor.2004.03.001

Stadtler, H. (2015). Supply chain management: An overview. In H. Stadtler, C. Kilger, & H. Meyr (Eds.), *Supply Chain Management and Advanced Planning* (pp. 3-28): Springer.

Steffen, T. (2001). *Modellierungsmethode zur Integration zwischenbetrieblicher Informationsflüsse*.

Sterman, J. D. (1989). Modeling Managerial Behavior: Misperceptions of Feedback in a Dynamic Decision Making Experiment. *Management Science, 35*(3), 321-339. doi:10.1287/mnsc.35.3.321

Sterman, J. D. (1994). Learning in and about complex systems. *System Dynamics Review, 10*(2-3), 291-330.

Stock, J. R., Boyer, S. L., & Harmon, T. (2010). Research opportunities in supply chain management. *Journal of the Academy of Marketing Science, 38*(1), 32-41. doi:10.1007/s11747-009-0136-2

Stölzle, W. (2005). Das Supply-Chain Operations Reference (SCOR)-Modell. *Controlling, 17*(8-9), 541-542.

Stölzle, W., & Karrer, M. (2003). Supply Chain Event Management-Impulse zur ereignisorientierten Steuerung von Supply Chains.

Stott, J., & Crosslin, R. (1992). *Modelling Information System Behavior: Integrating the Research Framework.* Paper presented at the Proceedings of the 3rd International Working Conference on Dynamic Modelling of Information Systems, Noordwijkerhout, June.

Strahringer, S. (1996). *Metamodellierung als Instrument des Methodenvergleichs: Eine Evaluierung am Beispiel objektorientierter Analysenmethoden*. Retrieved from

Strahringer, S. (2016). Modelle und Metamodellierung. *Geschäftsprozessorientierte Systementwicklung: Von der Unternehmensarchitektur zum IT-System*, 11.

Supply Chain Council, S. (2010). SCOR Supply Chain Operations Reference Model Version 10.0.

Tako, A. A., & Robinson, S. (2012). The application of discrete event simulation and system dynamics in the logistics and supply chain context. *Decision Support Systems, 52*(4), 802-815.

Taveter, K., & Wagner, G. (2001). *Agent-oriented enterprise modeling based on business rules.* Paper presented at the 20th Int. Conf. on Conceptual Modeling (ER2001).

Thomas, O. (2005). *Das Modellverständnis in der Wirtschaftsinformatik: Historie, Literaturanalyse und Begriffsexplikation*: Inst. für Wirtschaftsinformatik.

Thompson, S. V., & Riding, R. J. (1990). The Effect of Animated Diagrams on the Understanding of a Mathematical Demonstartion in 11- to 14-Year-Old Pupils. *British Journal of Educational Psychology, 60*(1), 93-98. doi:10.1111/j.2044-8279.1990.tb00925.x

TOVE Ontology Project. (2011a). Resource Ontology. Retrieved from http://www.eil.utoronto.ca/tove/resource/res52.html

TOVE Ontology Project. (2011b). Taxonomy of roles. Retrieved from http://www.eil.utoronto.ca/tove/resource/res52.html

Trost, M. (2008). *Gesamtheitliche Anlagenmodellierung und -analyse auf Basis stochastischer Netzverfahren.*

Tversky, B., Morrison, J. B., & Betrancourt, M. (2002). Animation: can it facilitate? *International Journal of Human-Computer Studies, 57*(4), 247-262. doi:http://dx.doi.org/10.1006/ijhc.2002.1017

Umeda, S. (2010). A simulation technology for supply-chain integration.

Umeda, S. (2013, 8-11 Dec. 2013). *Simulation analysis of supply chain systems with reverse logistics.* Paper presented at the 2013 Winter Simulations Conference (WSC).

Umeda, S., & Jain, S. (2004). Integrated Supply Chain Simulation System (ISSS)-Modeling Requirements and Design Issues. *National Institute of Standards and Technology, US Deparment of Commerce. Washington, DC[Links]*.

Umeda, S., & Lee, Y. (2004). *Design specifications of a generic supply chain simulator*. Paper presented at the Proceedings of the 36th conference on Winter simulation, Washington, D.C.

Umeda, S., & Zhang, F. (2006). Supply chain simulation: generic models and application examples. *Production Planning & Control, 17*(2), 155-166. doi:10.1080/09537280500224028

Umeda, S., & Zhang, F. (2008). Hybrid Modeling Approach for Supply-Chain Simulation *Lean Business Systems and Beyond* (pp. 453-460): Springer.

Van Der Zee, D. J., & Van Der Vorst, J. G. A. J. (2005). A Modeling Framework for Supply Chain Simulation: Opportunities for Improved Decision Making*. *Decision Sciences, 36*(1), 65-95. doi:10.1111/j.1540-5915.2005.00066.x

Van Horenbeek, A., Pintelon, L., & Muchiri, P. (2010). Maintenance optimization models and criteria. *International Journal of Systems Assurance Engineering and Management, 1*(3), 189-200. doi:10.1007/s13198-011-0045-x

Vasilecas, O., Kalibatiene, D., & Lavbič, D. (2016). Rule- and context-based dynamic business process modelling and simulation. *Journal of Systems and Software, 122*, 1-15. doi:http://dx.doi.org/10.1016/j.jss.2016.08.048

VDI. (1970). Begriffe und Erläuterungen im Förderwesen (Vol. VDI-Richtlinie 2411): VDI-Verlag.

VDI. (2014). VDI 3633-1 Simulation von Logistik-, Materialflussund Produktionssystemen *Grundlagen* (Vol. VDI 3633): VDI.

Vegetti, M., Gonnet, S., Henning, G., & Leone, H. *Towards a supply chain ontology of information logistics within process industry environments*.

Vegetti, M., Gonnet, S., Henning, G., & Leone, H. (2005). *Towards a supply chain ontology of information logistics within process industry environments*. Paper presented at the 4th Mercosur Congress on Process Systems Engineering, Rio de Janeiro.

Visic, N., Fill, H. G., Buchmann, R. A., & Karagiannis, D. (2015, 13-15 May 2015). *A domain-specific language for modeling method definition: From requirements to grammar.* Paper presented at the 2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS).

W3C. (2014). RDF 1.1 Concepts and Abstract Syntax. Retrieved from http://www.w3.org/TR/rdf11-concepts/

Wagenitz, A. (2007). Modellierungsmethode zur Auftragsabwicklung in der Automobilindustrie. *Dortmund, Univ., Diss., Düsseldorf*.

Wagner, G. (1998). *Foundations of knowledge systems : with applications to databases and agents*. Boston [u.a.]: Kluwer.

Wagner, G. (2002). *How to Design a General Rule Markup Language*. Paper presented at the Workshop XML Technologien für das Semantic Web (XSW 2002}.

Wagner, G. (2005). Rule Modeling and Markup. Reasoning Web. In N. Eisinger & J. Maluszynski (Eds.), (Vol. 3564, pp. 96-96): Springer Berlin / Heidelberg.

Wagner, G., Antoniou, G., Tabet, S., & Boley, H. (2004, 20-24 Sept. 2004). *The Abstract Syntax of RuleML - Towards a General Web Rule Language Framework.* Paper presented at the Web Intelligence, 2004. WI 2004. Proceedings. IEEE/WIC/ACM International Conference on.

Wang, H. (2002). A survey of maintenance policies of deteriorating systems. *European Journal of Operational Research, 139*(3), 469-489. doi:10.1016/s0377-2217(01)00197-7

Warnecke, H.-J., & Braun, J. (2013). *Vom Fraktal zum Produktionsnetzwerk: Unternehmenskooperationen erfolgreich gestalten*: Springer-Verlag.

Weilkiens, T. (2006). Systems engineering mit SysML/UML. *Modellierung, Analyse, Design. dpunkt. Verlag GmbH.*

Werner, H. (2013). *Supply Chain Management: Grundlagen, Strategien, Instrumente und Controlling*: Springer-Verlag.

Whitten, J. L., Bentley, L. D., & Dittman, K. C. (2004). Systems analysis and design methods: New York: McGraw Hill.

Wong, C. Y., McFarlane, D., Zaharudin, A. A., & Agarwal, V. (2002, 6-9 Oct. 2002). *The intelligent product driven supply chain.* Paper presented at the IEEE International Conference on Systems, Man and Cybernetics.

www.adoxx.org. ADOxxWEB Simulation. Retrieved from https://www.adoxx.org/live/adoxxweb-simulation-details

Ye, Y., Yang, D., Jiang, Z., & Tong, L. (2008a). An ontology-based architecture for implementing semantic integration of supply chain management. *International Journal of Computer Integrated Manufacturing, 21*(1), 1-18. doi:10.1080/09511920601182225

Ye, Y., Yang, D., Jiang, Z., & Tong, L. (2008b). Ontology-based semantic models for supply chain management. *International Journal of Advanced Manufacturing Technology, 37*(11-12), 1250-1260. doi:10.1007/s00170-007-1052-6

Zhao, X., Liu, C., & Lin, T. (2012). Incorporating business logics into RFID-enabled applications. *Information Processing & Management, 48*(1), 47-62.

Zülch, G., & Stock, P. (2010). Simulationsunterstütztes Wertstromdesign: Ansatz zur Steigerung des Wertschöpfungspotenzials in der Baustoffindustrie *Integrationsaspekte der Simulation: Technik, Organisation und Personal*: KIT Scientific Publishing, Karlsruhe.

11. Anhang

11.1 Zusammenfassung

Effiziente und reaktionsschnelle Supply Chain Networks bedingen eine enge Verknüpfung und verzögerungsfreie Kopplung von Informations- und Materialflüssen. Besonders wichtig ist das synchrone Zusammenspiel in der Domäne Instandhaltung. Die sogenannten Maintenance Supply Chains (MSC) müssen regelmäßige Wartungsaufgaben ressourcenschonend durchführen aber gleichzeitig in der Lage sein, bei ungeplanten Ereignissen, wie zum Beispiel Maschinenausfällen, kurzfristig, flexibel und zielgerichtet zu reagieren. In einer MSC werden Informationen über den Zustand der Anlage oder Ersatzteilbestände zur Steuerung herangezogen und erzeugen als Aufträge wiederum Informationsflüsse, welche auf die Materialbewegung einwirken.

Diese komplexen und hochdynamischen Wechselwirkungen können mit mathematischen Modellen oder klassischen Geschäftsprozessmodellierungsmethoden, die auf den Kontrollfluss ausgerichtet sind, nicht transparent gemacht werden. Bekannte Materialfluss Simulatoren berücksichtigen den Informationsfluss unzureichend, gefärbte Petri Netze hingegen stellen den jeweiligen Systemzustand modellhaft gut dar, werden aber aufgrund der Beschränkung der Modellierungsklassen für typische Problemstellungen schnell unübersichtlich groß.

Das Dissertationsprojekt hat zum Ziel eine domänenspezifische, graphische Modellierungsmethode zu entwickeln welche die Dokumentation, Analyse und Verbesserung von Maintenance Supply Chains unterstützt. Bestandteile der Modellierungsmethode sind die Modellierungssprache, Auswertungsmechanismen und Algorithmen und das Vorgehensmodell. Die Modellierungssprache wird unter Berücksichtigung von Konzepten des SCOR-Modells (Supply Chain Reference Modell) und von Basisklassen bekannter Supply Chain Ontologien entwickelt. Die statische Struktur der Supply Chain wird graphisch modelliert und das aktive und reaktive Verhalten ihrer Objekte mittels Regeln beschrieben. Ein zu entwickelnder Simulationsalgorithmus dynamisiert das Modell und generiert sogenannte Zustandsfolgediagramme. Diese machen die chronologischen Wirkungszusammenhänge von Steuerungsinformationen und resultierenden Materialbewegungen im logistischen Netzwerk transparent. Das Vorgehensmodell enthält Anweisungen für den Modellierer zur effektiven Nutzung der Methode.

Die Methode wird formal beschrieben und auf der Metamodellierungsplattform ADOxx prototypisch entwickelt. Der Ansatz wird auf Basis von praxisnahen Beispielen evaluiert und der Prototyp nach Abschluss und bei Bedarf mittels der Plattform Open Models Laboratory (OMiLAB) der wissenschaftlichen Community zur Verfügung gestellt.

11.2 Abstract

Efficient and responsive supply chain networks require a close connection and lag-free coupling of information and material flows. Particularly important is the synchronous interaction in the maintenance domain. So called Maintenance Supply Chains (MSC) have to perform regular maintenance tasks resource efficient and at the same time be able to respond quickly, flexibly and purposefully in case of unplanned events, such as machine failures. Information on the condition of an asset or on stock of replacement parts are used to control an MSC and produce orders which act as information flows on the material movement. These complex and highly dynamic interactions can't be made transparent with mathematical models or classic business process modeling methods that are aligned with the control flow. Known material flow simulators consider the information flow insufficiently. Colored Petri Nets, however represent the respective system state well but quickly become confusing large for typical problems due to the limitation of the modeling classes.

This PhD thesis project aims to develop a domain-specific, graphical modeling methodology that supports the documentation, analysis and improvement of Maintenance Supply Chains. Components of the modeling method are the modeling language, evaluation mechanisms and algorithms and the modelling procedure. The modeling language is developed in consideration of concepts of the SCOR model (Supply Chain Reference Model) and of base classes of known Supply Chain ontologies. The static structure of the supply chain is graphically modeled and describes the active and reactive behavior of its objects by rules. A to be developed simulation algorithm analyses the dynamics of the model and generates so-called state sequence models. These make the chronological interrelationships of control information and the resulting material movements in the logistic network transparent. The modelling procedure includes instructions for the modeler to make effective use of the method.

The method is described formally and developed as a prototype on the Meta modelling platform ADOxx. The approach will be evaluated on the basis of practical examples and after completion the prototype can be provided to the scientific community via the platform Open Models OMiLAB.