DOI 10.2195/LJ_PROC10_Rehm

Ein Metamodell von Produktionssystemen als Grundlage für die automatische Simulationsmodellgenerierung

Dipl.-Wi.-Ing. Markus Rehm*, Dipl.-Inf. Oliver Schönherr**,
Prof. Dr.-Ing. habil. Thorsten Schmidt*
Technische Universität Dresden
**Professur für Technische Logistik, **Professur für Modellierung und Simulation*

Abstract: Ein Beitrag zur Schaffung einer effizienten und flexibel einsetzbaren Simulationsumgebung für die Termin-, System- und Ressourcenplanung logistischer Prozesse und Produktionssysteme, ist die von einem spezifischen Simulationssystem unabhängige Erzeugung der Simulationsmodelle. Die hierfür an der Technischen Universität Dresden entwickelte informationstechnische Architektur, beinhaltet ein Metamodell zur methodischen Unterstützung der deskriptiven Abbildung und Modellierung diskreter Prozesse einschließlich deren notwendigen Interdependenzen. Es liegt die Annahme zugrunde, dass sich derartige Systeme in die Bereiche Struktur, Verhalten und Steuerung unterteilen lassen. Der Artikel stellt diese grundlegenden Bestandteile im Kontext der automatischen Simulationsmodellgenerierung vor und erläutert die hierfür erforderlichen Voraussetzungen und Beschreibungsmethoden einführend.

1 Einleitung

Um komplexe Systeme in all ihren Ausprägungen und Aspekten umfassend klassifizieren, beschreiben und modellieren zu können, ist es zunächst hilfreich deren Facetten im Sinne der Dekomposition in abgrenzbare Strukturen zu unterteilen und lokale wie globale Wirkmechanismen zu erfassen. In diesem Zusammenhang ist die Verfügbarkeit eines Werkzeuges elementar wichtig, welches ein derartiges Vorgehen auf der Grundlage seiner inhärenten Merkmale methodisch unterstützt und dabei dem Anspruch der allgemeinen Anwendbarkeit genügt. Hierfür bedarf es somit einer einheitlichen, ausreichend mächtigen und möglichst freien Modellierungssprache, mit deren Hilfe alle notwendigen Aspekte und Daten von Systemen abzubilden sind. Dies bildet sodann die Grundlage der automatischen Simulationsmodellgenerierung, wodurch die Möglichkeit gegeben wird, Systeme und Szenarien losgelöst von einem konkreten Simulationswerkzeug zu modellieren. Auf diesem Wege ist neben der Austauschbarkeit der Simulationsmodelle zudem ein Ansatz für eine herstellerunabhängige Sicht auf Systeme greifbar, was zugleich die Grenzen proprietärer Simulatoren überwinden lässt**.**

1.1 Stand der Technik und Ist-Situation

Im Gegensatz zur Softwaretechnik konnte sich für den Bereich des Systems Engineering bislang keine einheitliche Sprache zur Modellierung diskreter Prozesse durchsetzen [WEI06][Fow03]. Dieses Problems nahm sich die Object Management Group (OMG) an und veröffentlichte im September 2007 die auf der Unified Modeling Language (UML) 2 basierende Systems Modeling Language (SysML) 1.0, welche sich als standardisierte Beschreibungssprache zur grafischen Modellierung und Abbildung komplexer Systeme behaupten soll. Ziel dabei ist es, Entwurf, Analyse und Verifikation von Systemen basierend auf den Prinzipien des model-based systems engineering (MBSE) entsprechend zu unterstützen und zu vereinfachen. Hierbei besteht der Fokus auf einer klaren und eindeutigen Beschreibung der Struktur, des Verhaltens und der Funktionalität von Systemen, welche durch die Zerlegung in statische und dynamische Bestandteile umfassend abgebildet werden können [SKHH06]. Doch bietet sich ausgehend von diesen formalen und objektorientierten Beschreibungen auch die Möglichkeit der Ableitung ausführbarer Modelle, was schließlich als eine Basis für die automatische Simulationsmodellgenerierung mit ihren vielfältigen Vorteilen angesehen werden darf [PJ08].

1.2 Zielstellung

Ziel der Forschungsarbeit ist es, dem Anspruch der automatischen Simulationsmodellgenerierung dahingehend gerecht zu werden, simulatorunabhängige, deskriptive Meta-Modellstrukturen unter Zuhilfenahme einer weitestgehend domänenunspezifischen Beschreibungssprache und der nötigen informationstechnischen Architektur zu erzeugen. Um dieser Zielstellung gerecht zu werden, gilt es ein allgemeines Lösungskonzept für die Modellierung von Optimierungsaufgaben der Termin-, System- und Ressourcenplanung zu entwickeln, in dem sich alle hierfür relevanten Aspekte diskreter Systeme wiederfinden. Die Modelle werden mit SysML entworfen, womit zugleich eine bestmögliche Ausgangslage für die spätere Überführung in verschiedene Simulatoren aus dem Bereich der Produktion und Logistik geschaffen wird.

2 Architektur der automatischen Modellgenerierung

Um das Ziel der automatischen und simulatorunabhängigen Erzeugung von Simulationsmodellen erreichen zu können, bedarf es einer entsprechenden informationstechnischen Architektur. Hierbei sollen die mit einem Modellierungswerkzeug und unter Vorgabe der Metamodelle erstellten SysML-Modelle

in Modelle für beliebige Simulationswerkzeuge transformiert werden. Zuvor ist das SysML-Modell mittels eines Parsers in ein internes Modell zu überführen, in dem ausschließlich die wichtigen und für die weitere Verarbeitung als notwendig erachteten Informationen gefiltert und strukturiert vorliegen. Anschließend kommt ein Translator-Plugin zum Einsatz, welches die Daten des internen Modells für ein Modell eines spezifischen Simulationswerkzeuges anhand der dort vorherrschenden Regeln und Formate aufbereitet. Während also das interne Modell allein an das Daten-Metamodell angelehnt ist und daher nur einmal erzeugt werden muss, bedarf es für jeden Simulator eines gesonderten Translator-Plugins (Abbildung 1: Architektur).

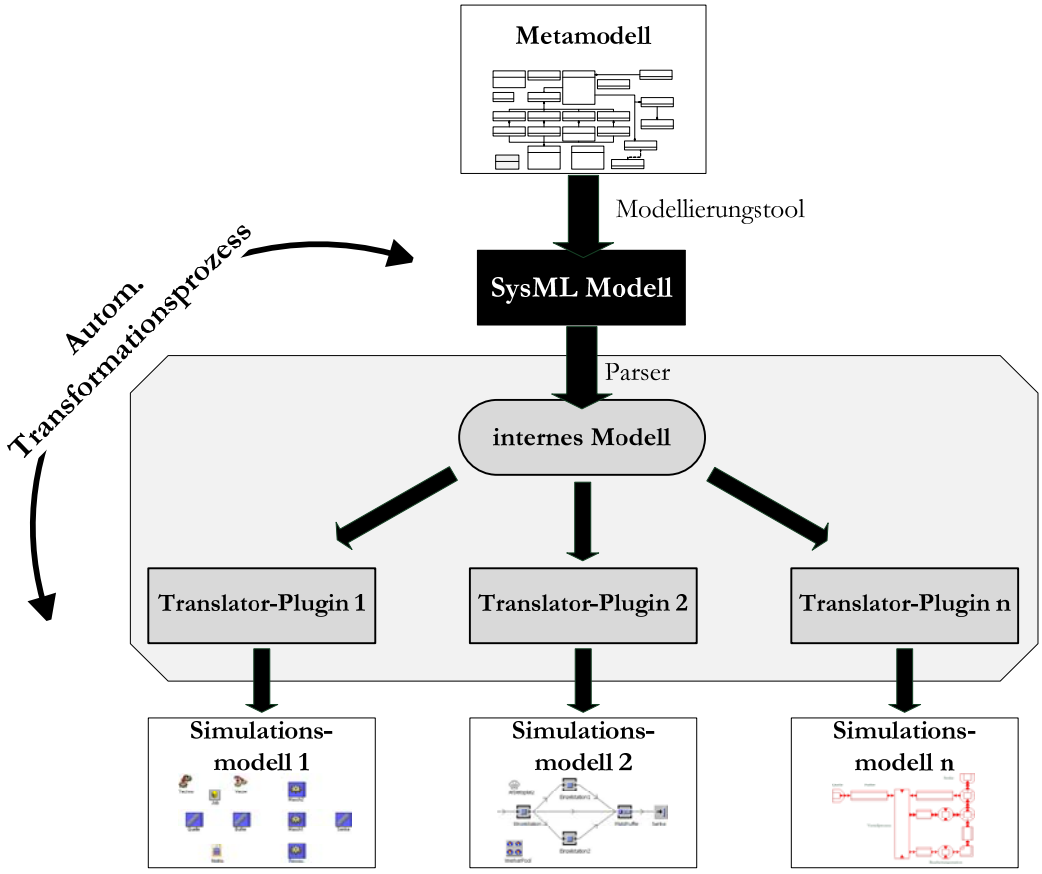

Abbildung 1: Architektur der automatischen Simulationsmodellgenerierung

Dies stellt prinzipiell einen Ausgangspunkt dar, um die Vorzüge der Metamodellierung mit der automatischen Erzeugung simulatorspezifischer Modelle zu verbinden. Zugleich ist damit der umgekehrte Weg möglich, bei dem aus spezifischen Modellen von Simulationswerkzeugen unter Berücksichtigung der dortigen Eigenheiten SysML-Modelle generiert werden. So kann unter Zuhilfenahme obiger Architektur eine beliebige Transformation zwischen Modellen unterschiedlicher Simulatoren und den SysML-Modellierungswerkzeugen stattfinden, womit über diesen Zwischenschritt gleichzeitig eine Austauschbarkeit zwischen den Simulatoren und ihren jeweiligen Modellen möglich ist.

3 Bestandteile des Metamodells

Elementarer Bestandteil der vorgestellten Systemarchitektur und gleichzeitig inhaltlicher Schwerpunkt des vorliegenden Artikels ist das Verhaltens-, Steuerungs- und Strukturmodell. Simulationssysteme gliedern Modelle implizit in statische und dynamische Bereiche auf, womit zugleich eine Grundvoraussetzung für die allgemeine SysML Modellbildung als existent angesehen werden kann. Während der strukturelle Teil die statischen Strukturen eines Systems bestehend aus Objekten und ihren Beziehungen repräsentiert, sind im Verhaltensmodell die dynamischen Prozesse eines Systems hinterlegt. Im Steuerungsmodell schließlich werden Routinen für das Dispatching und Routing im Sinne übergeordneter situativer Entscheidungen und Anweisungen dargestellt.

3.1 Das Verhaltensmodell

Um das dynamische Verhalten der Objekte selbst und untereinander zu beschreiben, ist die Identifikation elementespezifischer Verhaltensmuster erforderlich. Ein derartiges Muster kann beispielsweise für einen Puffer daraus bestehen, dass das jeweilige Element auf einzelne Werkstücke oder Fertigungslose wartet, diese anschließend einordnet (und ggf. vorher sortiert), um es danach (ggf. bedingt) weiterzuleiten. Derartigen Verhaltensmustern lassen sich übergeordnete Verhaltensphasen zuordnen, die im Falle eines Pufferelementes das Entgegennehmen, Einordnen und Weiterleiten sind.

Das Verhalten der Elemente und (Teil-)Abschnitte von Produktionssystemen kann hierarchisch verschiedenartig in Granularitätsstufen untergliedert werden. Auf der *Prozessebene* erfolgt die Modellierung der konkreten Abfolge eines Produktionsprozesses, wie es beispielsweise der Arbeitsplan vorsieht. Bestimmender Faktor des Verhaltens ist hierbei der eigentliche Objektfluss. Auf der darunter liegenden *Verhaltensebene* finden sich oben angedeutete Verhaltensmuster wieder, die die jeweiligen Phasen, die ein Objekt auf einem Element durchläuft, charakterisieren. Da das Verhalten auf dieser Ebene streng deterministisch verläuft, ist es der Kontrollfluss, welcher hier den maßgeblichen Faktor darstellt. Konkrete Aktionen und Prozessschritte werden schließlich auf der *Ausführungsebene* beschrieben, wo eine feingranulare Spezifikation der Vorgänge stattfindet. Somit können der Informationsfluss und die regelbasierten Anweisungen der operativen Produktionssteuerung in diesem Bereich als besonders wichtig angesehen werden.

Des Weiteren ist es gerade aus Sicht der Modellierung von Interesse, inwiefern die Reihenfolge der Vorgänge a priori determiniert sowie deren Einhaltung elementar wichtig ist. Abhängig davon ist schließlich die Wahl der SysML

Diagrammform, mit welcher eine derartige Verhaltensstruktur abzubilden ist – Aktivitätsdiagramme eignen sich für die Darstellung von Reihenfolgebeziehungen, Zustandsdiagramme werden für reihenfolgeunabhängige Vorgänge bevorzugt (Abbildung 2: Modellierung der Verhaltens).

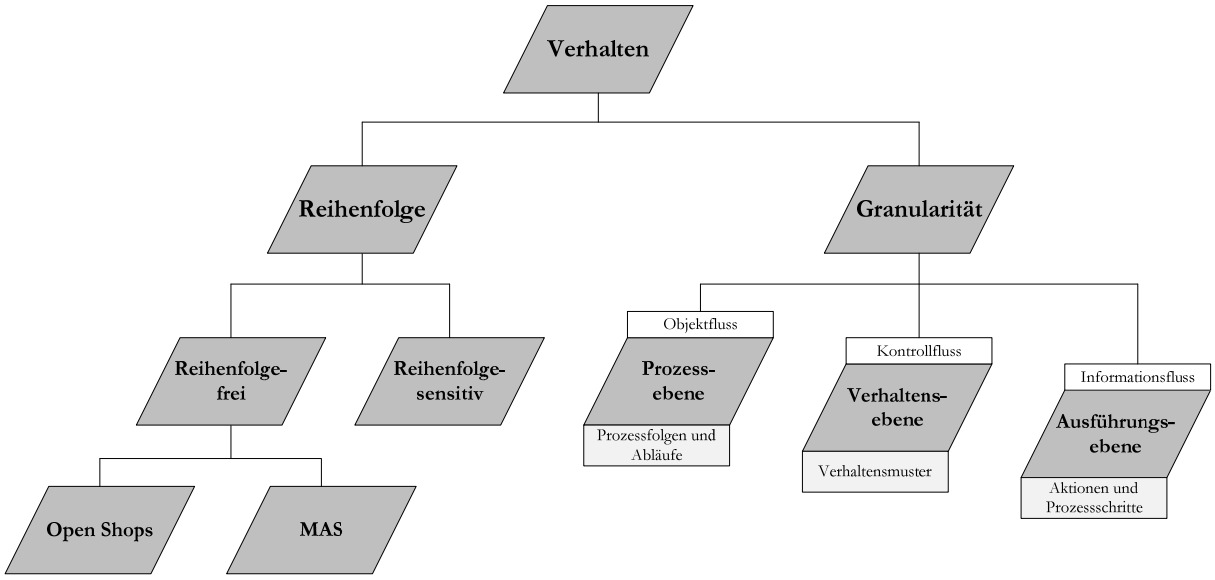

Abbildung 2: Modellierung des Verhaltens

Ein Vorteil dieser Kapselung von Verhaltensweisen ist die Wiederverwendbarkeit anzutreffender Stereotypen bzw. Teilstücke, womit der Modellierungsaufwand verringert werden kann. Insbesondere im Bereich der Verhaltens- und Ausführungsebene existieren allgemeine und sich wiederholende Schemata, die zudem hierarchisch aufeinander aufbauen. Den Verhaltensphasen der Prozessebene können Verhaltensmuster aus der Verhaltensebene zugeordnet werden, diese wiederum basieren auf Aktionsmustern der Ausführungsebene.

3.2 Das Steuerungsmodell

Grundlegende Bestandteile der im zweiten Abschnitt vorgestellten Systemarchitektur sind das Struktur- und Verhaltensmodell. Darüber hinaus gilt es zu untersuchen, in welcher Art und Weise das Steuerungssystem aufgebaut und eingebunden werden kann. Wie bei einer Vielzahl der analysierten Simulationssysteme aus dem Bereich Produktion und Logistik festzustellen war, befinden sich die hierfür notwendigen Algorithmen (Prioritäten, Reihenfolgevorgaben etc.) üblicherweise direkt in den Systembausteinen selbst, wodurch das eigentliche Basissystem mit dem Steuerungssystem funktional verschränkt ist. Ziel sollte es sein, dieses Paradigma vor dem Hintergrund der Metamodellierung und automatischen Simulationsmodellerzeugung zu durchbrechen. In diesem Zusammenhang gilt es die Schnittstellen, Elemente sowie deren Funktionen und erforderlichen ein- und ausgehenden Informa-

tionen zusammenhängend zu aggregieren, womit eine objektorientierte Analyse zu begründen ist.

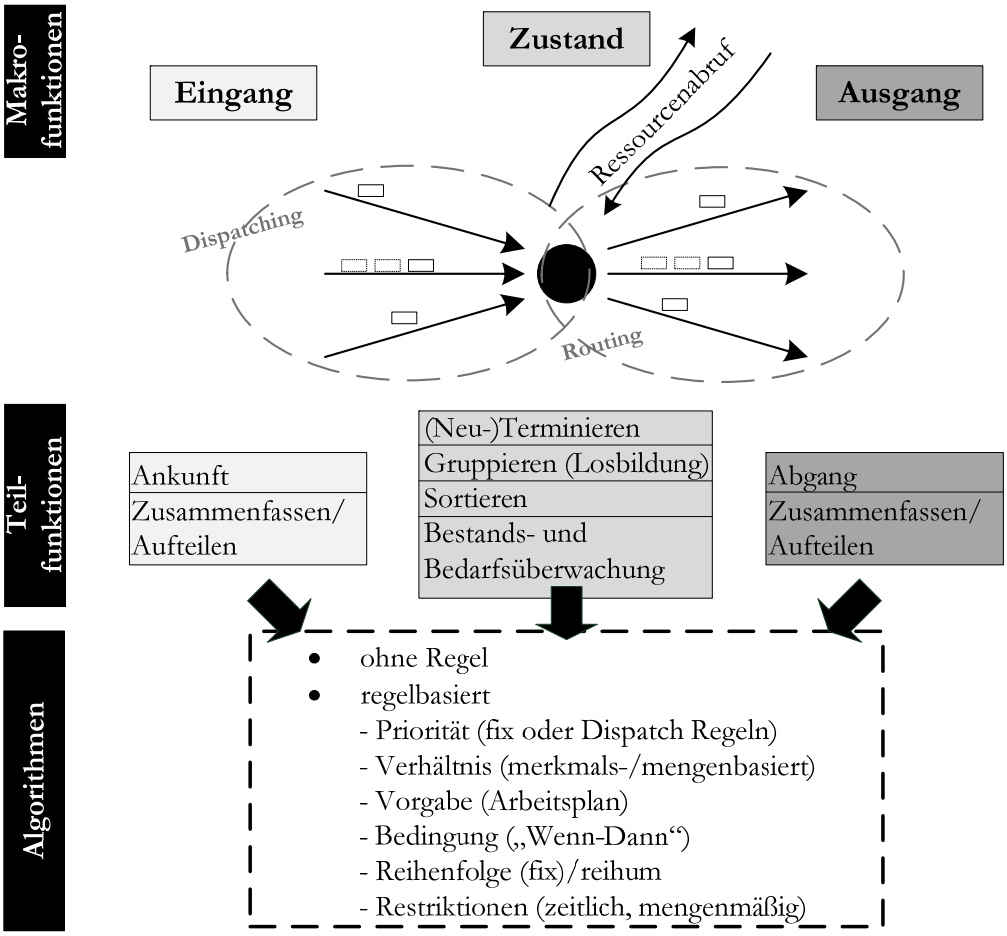

Abbildung 3: Funktionalitäten an Modellobjekten

Entsprechend obiger Abbildung 3: Funktionalitäten, sind den jeweiligen Makrofunktionen Teilfunktionen zugeordnet. Insbesondere durch Ankunfts- und Abgangsfunktionen ist es möglich, zeitliche oder auch mengenmäßige Beschränkungen zu implementieren. In gleicher Weise kann zudem der Ressourcenabruf erfolgen, sollte es keine feste Zuordnung zu einem Prozess geben. Sortier- und Gruppieralgorithmen finden innerhalb des Prozesses selbst statt, insofern sie vom eigentlichen Eingangsprozess loszulösen sind (was durchaus bei speziellen Warteschlangenelementen der Fall sein kann). Handelt es sich um einen vorwärtsgerichteten Ablauf, wie dies in Abbildung 3 der Fall ist, so darf an dieser Stelle eine Orientierung hinsichtlich Dispatching und Routing gegeben werden. Insbesondere dann, wenn ein Knoten auf der Eingangsseite aktiv geschaltet ist, erfolgt die Priorisierung und Reihenfolgebildung lokal (jedoch durchaus auf der Basis globaler Informationen) und regelbasiert (und nicht zwangsläufig durch eine übergeordnete Instanz). Betrachtet werden dann ausschließlich der Teilgraph

selbst und die zeit- oder ereignisorientiert aufgerufenen Funktionen (Ankunft/Zusammenfassen). Es ist es zudem vorstellbar, dass Anfragen/-Aufträge durch die Betriebsdatenerfassung (Ermittlung Zwischenankunftszeiten, Bestandsführung etc.) und einen Vergleich der Bedarfe und Angebote einer Station, ausgelöst werden. Weiterhin bzw. in Folge dessen kann durch Kommunikation innerhalb des Systems ein Kapazitätsabgleich stattfinden. Zu erwähnen ist außerdem, dass auch bei vorwärtsgerichteter Betrachtungsweise durch die Eingangsfunktion eine Aufteilung eingehender Objekte durchgeführt werden kann, wenn beispielsweise eine losweise Bearbeitung ausgeschlossen werden soll. An dieser Stelle sei angemerkt, dass die Orientierungsrichtung des Steuerungsablaufes insbesondere bei der (Neu-)Terminierung von Aufträgen eine Rolle spielt. So können wie üblich späteste Startzeitpunkte bei Rückwärtsbetrachtung und früheste Endzeitpunkte bei Vorwärtsbetrachtung ermittelt werden. Ist der Ablauf rückwärtsorientiert, handelt es sich an Knotenausgängen gewöhnlich um Bedarfe (entsprechend um Angebote bei Vorwärtsorientierung), die beispielsweise im Falle der Montage eine Stücklistenauflösung zur Folge haben, was wiederum auf vorgelagerte Prozesse Auswirkungen (z.B. das Auslösen von Fertigungsaufträgen) hat.

Für die Abbildung des Steuerungsmodells gibt es im Gegensatz zum Struktur- und Verhaltensmodell keine Modellierungsansätze. Doch stellen die beiden Letztgenannten das bereits angesprochene Basissystem dar, weshalb eine Schnittstellenanalyse ein unabdingbarer Schritt für eine ganzheitliche Metamodellierung ist. Da das Steuerungsmodell selbst aus Struktur- und Verhaltensmodellen besteht sowie SysML als Modellierungssprache entsprechende Schnittstellen bereitstellt, lässt sich das Steuerungsmodell mit den SysML immanenten Mitteln umsetzen. Diese Thematik stellt insbesondere den Inhalt der aktuellen Forschungsarbeit dar, weshalb an dieser Stelle auf weiterführende Erläuterungen verzichtet werden darf.

3.3 Das Strukturmodell

Für einen Flow Shop kann die Struktur eines Metamodells mit Hilfe von SysML Blockdefinitionsdiagrammen abgebildet werden [HRM07]. Um jedoch ein möglichst allgemeines Strukturmodell zu erstellen, müssen zunächst die Objekte eines Systems sowie deren Beziehungen und Attribute identifiziert werden. Grundsätzlich kann man zwischen *imaginären* und *realen* Objekten sowie *Hilfsobjekten* differenzieren. Im Zusammenhang mit formalisierten allgemeinen Strukturen stellen imaginäre Objekte Prozesse dar, welche ihrerseits in Bearbeitungs-, Ankunfts-, Abgangs- und Warte-/Lagerprozesse unterteilt werden können. Reale Objekte (Bauteile, Material etc.) betreten ein

aus diesen Prozessen definiertes Bediensystem und erfahren innerhalb dessen Zustands- oder Ortsänderungen. Die hierfür erforderlichen Ressourcen (Mitarbeiter, Betriebsmittel etc.) sind neben den Hilfsobjekten ein domänenspezifischer Faktor, da die Annahme getroffen wird, dass imaginäre Objekte in jeglichen diskreten Systemen vorzufinden sind. Hilfsobjekte ihrerseits sind keine hinreichend notwendige Objektklasse, können jedoch die Modellbildung erheblich verbessern. Als Beispiel sind an dieser Stelle Modi zu nennen, wodurch zum Ausdruck kommt, mit welcher Ressourcenkombination ein bestimmter Prozess ausführbar ist. Die dahinter durch Attribute definierten Konfigurationen bestimmen schließlich die Faktorkombination, womit im Zuge der Planung ein optimaler Durchlauf erzeugt werden soll, welche Modi also in Abhängigkeit von der gewählten Zielfunktion zu favorisieren sind. Relationen und Dependenzen zwischen Objekten werden durch Beziehungen näher bestimmt. Sind diese mit Bedingungen oder Attributen verknüpft, kann von komplexen Beziehungen gesprochen werden, wie dies beispielsweise im Fall spezieller Freigaberegeln oder Haltebedingungen denkbar ist. Einfache Beziehungen andererseits sind übliche Trivialfälle wie die Reservierung einer Ressource durch einen Prozess, weshalb diese Relationen nicht von der eigentlichen Anwendungsdomäne abhängig sind (Abbildung 4: Objekt- und Beziehungsklassen).

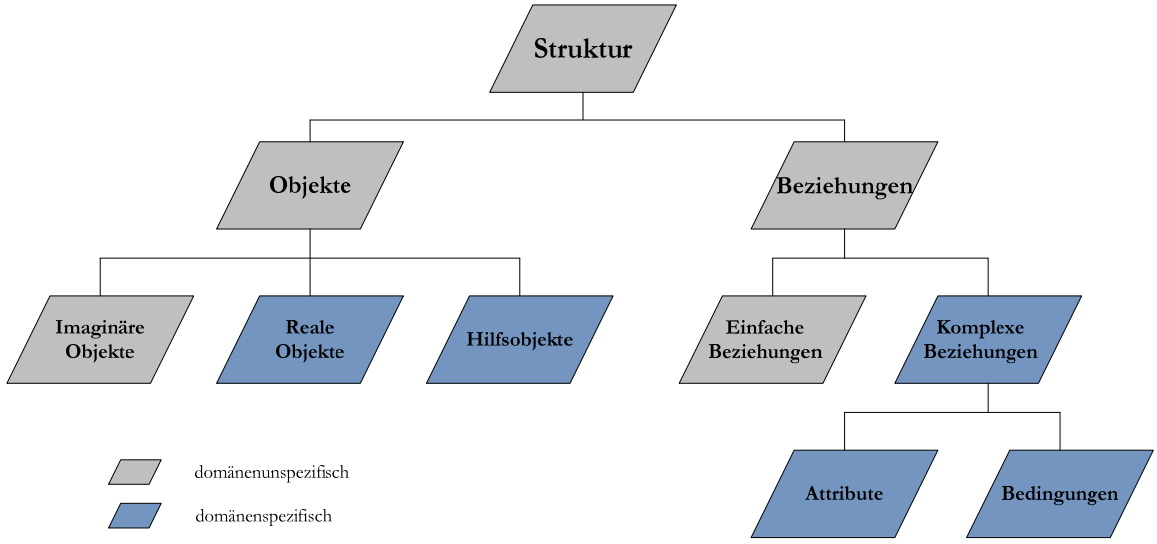

Abbildung 4: Objekt- und Beziehungsklassen des Strukturmodells

Das eigentliche strukturelle Metamodell der Produktionsdomäne kann im Sinne der Modellierung auf einer um den Faktor eins niedrigeren Abstraktionsstufe angesiedelt werden. Ohne an dieser Stelle im Detail auf Modellierungsart und die hierfür notwendigen Tools sowie Methoden einzugehen, sollen in diesem Rahmen zumindest die relevantesten Elemente des Modells ein-

schließlich zugehöriger Attribute Erwähnung finden, um einen generellen Einblick in den Modellaufbau zu gewähren.

Wie erläutert, sind so genannte Vorgangsmodi stets dann von Interesse, wenn ein Prozess mit unterschiedlichen Ressourcenkombinationen ausgeführt werden kann. Daher wird im Modell eine dem Prozess zugeordnete Klasse Modi mit den zugehörigen Attributen *processingTime* und *priority* hinterlegt. Darüber hinaus werden Aufträge und die jeweiligen Produkte, welche ihrerseits Prozesse konsumieren und damit zugleich Ressourcen binden bzw. verbrauchen, definiert. Mithilfe von Eigenschaften wie *earliestStart*, *latestEnd*, *description* und *priority* werden diese weiter spezifiziert. Da sich eine Vielzahl kommerzieller Simulationssysteme aus dem Bereich Produktion und Logistik Pools für Maschinen, Transportmittel oder Mitarbeiter bedient, darf hierfür eine explizite Abbildung nicht fehlen, auch wenn in wieder anderen Systemen jegliche Ressourcen im Sinne von Prozessen modelliert werden (Abbildung 5: Strukturelles Metamodell).

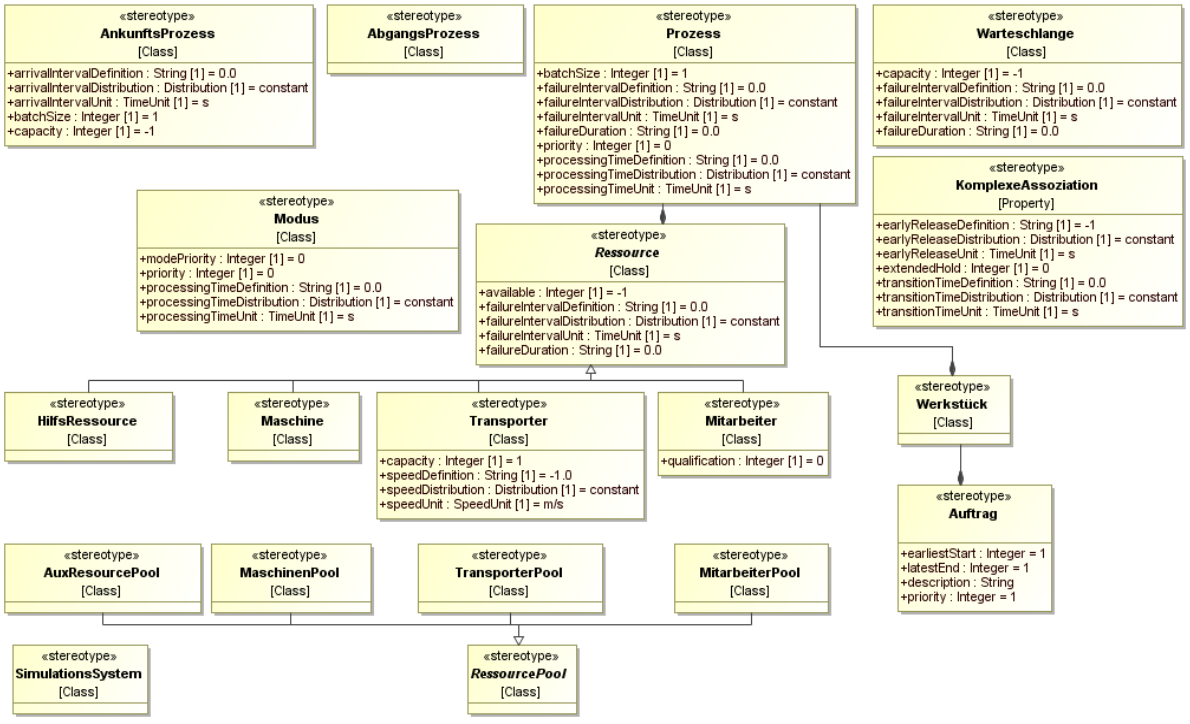

Abbildung 5: Strukturelles Metamodell der Produktionsdomäne

Insbesondere beim Aufbau des strukturellen Metamodells ist die Domänenspezifik von besonderem Interesse. Produktionssysteme besitzen partiell andere Modellierungselemente als beispielsweise ein Krankenhaus, eine Baustelle, eine Großküche oder Systeme der technischen Logistik. Allen gemein ist allein das Vorhandensein imaginärer Objekte sowie spezifischer realer und Hilfsobjekte, wie auch Ressourcen in dieser oder jener Form. Bezogen auf die Logistik etwa spielen Modi eine tendenziell untergeordnete

Rolle, Ressourcen hingegen sind vielfältig in Art und Anzahl vorhanden. Vor dem Hintergrund der Metamodellierung ist es daher unabdingbar, die grundlegenden logistischen Elemente nicht nur zu identifizieren sondern sinnvoll zu systematisieren und in einen gemeinsamen Kontext zu bringen.

So ist ein logistisches System wie eine Mehrzahl aller Systeme als eine Netzwerkstruktur bestehend aus Knoten und Kanten aufzufassen. In einem solchen System finden Transport-, Lagerungs-, Be- und Entlade-, Ein- und Auslager- sowie Kommissionierprozesse statt. Aus abstrakter Sicht können daher Raumüberbrückung (Transportieren), Zeitüberbrückung (Lagern) und Veränderung der Anordnung (Umschlagen und Kommissionieren) als wesentliche Merkmale derartiger Systeme aufgefasst werden. Schließlich bedarf es noch Befähigern zur Umsetzung, welche in diesem Zusammenhang das Ressourcenportfolio abbilden. Personen, Fördermittel, Lagertechnik und Informationen können hierfür genannt werden. Die beiden Erstgenannten sind als reale Objekte domänenspezifisch zu modellieren, ebenso die im System befindlichen Sachgüter, welche das durch das Netz laufende entity in Form des Transportgutes darstellen. Informationen gelten als immanente Bestandteile eines jeden Systems und sind des Verhaltensmodells zuordenbar.

Die Ressourcen werden auf Grundlage ihrer Attribute verschiedenartig gegliedert. Die übliche Einteilung in Stetig- und Unstetigförderer wird beibehalten, eine Einteilung in statische und dynamische Fördermittel jedoch im Kontext der strukturellen Modellierung unbeachtet gelassen. Unstetige Fördermittel sind u.a. mit Attributen wie Beschleunigung (*acceleration*), max. Geschwindigkeit (*speed*), und max. Ladekapazität (*capacity*) auszustatten. Stetige Fördermittel andererseits werden durch für sie relevante Eigenschaften wie Streckenlänge (*trackLength*), Förderabstand (*transferInterspace*) oder mittlere Fördergeschwindigkeit (*averageSpeed*) spezifiziert.

Darüber hinaus werden Lagerressourcen benötigt. Diese weisen als Hauptattribut für das hier beschriebene Hauptattribut lediglich eine bestimmte Kapazität (*capacity*) auf, welche implizit weitere Eigenschaften wie Fläche oder Anzahl der Lagermodule, -reihen, -spalten, -stapelfaktor etc. enthält. Daher erfolgt die Abbildung von Lagerressourcen in der Form von Warteschlangen, welche aus Sicht der Metamodellierung ein Synonym darstellen. Mitarbeiter und ihre entsprechende Qualifikation sind außerdem zu berücksichtigen. Ebenso wesentlich sind Förderhilfsmittel mit ihren jeweiligen Kapazitäten, können jedoch ebenso als eines der obigen Elemente modelliert werden.

Die gegenseitigen Assoziationen werden durch die Verknüpfungen im Blockdefinitionsdiagramm selbst wiedergegeben. Beziehungen, welche komplexerer Natur sind, sind hingegen gesondert zu definieren ("KomplexeAssoziation"). Im Falle eines logistischen Systems sind an dieser Stelle Freigabeanweisungen (*releaseDefinition*), Sicherheitsbestände (*inventoryLevel*) oder Kommissionier-/Sortierregeln (*sortDefinition*) hinterlegt. Ein Auszug des domänenspezifischen Struktur-Metamodells am Beispiel logistischer Systeme, kann durch die nachfolgende Abbildung 6: Strukturelles Metamodell gegeben werden.

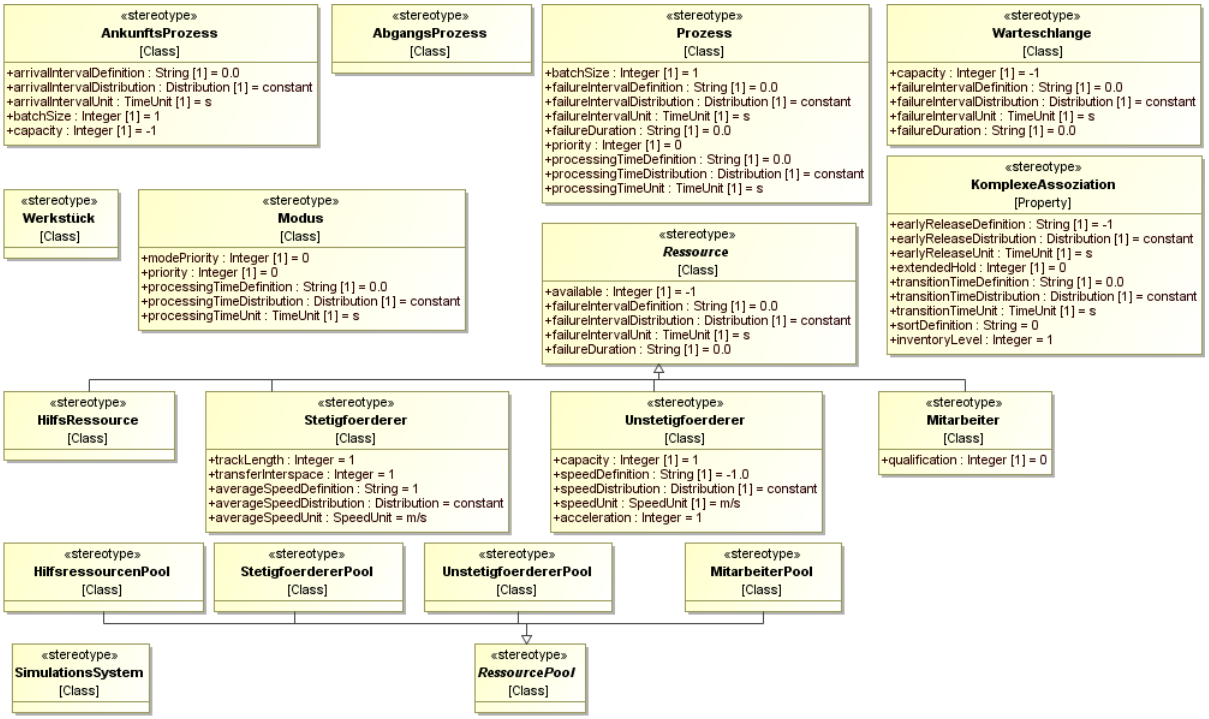

Abbildung 6: Strukturelles Metamodell für ein logistisches System (Auszug)

4 Zusammenfassung und Ausblick

Es wurde versucht, der Modellierung von diskreten Prozessen vor dem Hintergrund der automatischen Erzeugung von Simulationsmodellen für die Ressourcen-, Termin- und Kapazitätsplanung einen theoretischen Rahmen zu geben und deren signifikante Eigenschaften zu analysieren und zu strukturieren (vgl. Kapitel 3). Nicht alle Elemente, Verknüpfungen und Eigenheiten sollten und konnten in diesem Rahmen hinreichend ausführlich und vollständig identifiziert und erläutert werden, da auch gerade die Steuerung derartiger Systeme in der Regel sehr komplex ist. Der Entwurf von Produktionsszenarien und deren Übersetzung in verschiedene Simulationswerkzeuge konnte bereits erfolgreich umgesetzt werden. In dem vorliegenden Beitrag wurde verdeutlicht, dass die Domänenspezifik ein nicht zu verachtender Faktor ist, grundlegende Klassifikationen (vgl. Objekte im Strukturmodell) und wiederkehrende Muster

(vgl. Verhaltensmodell) jedoch einen vielversprechenden Weg zu einer effizienten Prozessmodellierung aufzeigen.

Weiterhin sind Problemstellungen aus anderen Domänen, wie der Krankenhauslogistik oder dem Bauwesen interessant. Durch eine abstrakte Sichtweise und einer großen Anzahl von Modellierungsmöglichkeiten, ist der Gebrauch von SysML aus Anwendersicht als schwierig zu bewerten. Aktuell wird daher an einem auf TOPCASED basierendem Modellierungstool, das den Ingenieur in seinen Anforderungen unterstützt, gearbeitet. Neben der Aufbereitung der allgemein modellierten Szenarien für verschiedene Simulationswerkzeuge, erfolgt zudem die Entwicklung eines SysML-Simulators. Dieser kann schließlich ohne Übersetzung direkt die mit SysML erstellten Modelle abbilden und behebt einige grundlegende Problempunkte wie die Einheit von Modellbaustein und Steuerungsalgorithmus.

Literatur

[Fow03] Fowler, Martin: UML konzentriert: Eine kompakte Einführung in die Standardobjektmodellierungssprache. Heidelberg: Pearson Education, 2003.

[HRM07] Huang, Edward; Ramamurthy, Randeep; McGinnis, Leon F.: System and Simulation Modeling using SysML. In: Proceedings of the 39th conference on Winter simulation: 40 years! The best is yet to come. Miami: Winter Simulation Conference, 2007.

[PJ08] Paredis, Christiaan J.J.; Johnson, Thomas: Using OMG's SysML to support simulation. In: Proceedings of the 40th Conference on Winter Simulation. Miami: Winter Simulation Conference, 2008.

[SKHH06] Scholz-Reiter, Bernd; Kolditz, Jan; Hildebrandt, Torsten; Höhns Hartmut: Modellierung selbststeuernder logistischer Prozesse in der Produktion. In: Karagiannis, D., Rieger, B. (Hrsg.). Herausforderungen in der Wirtschaftsinformatik. Berlin: Springer, 2006.

[Wei06] Weilkins, Tim: Systems Engineering mit SysML/UML: Modellierung, Analyse, Design. Heidelberg: dpunkt, 2006.