Sven Jensen

Eine Methodik zur teilautomatisierten Generierung von Simulationsmodellen aus Produktionsdatensystemen am Beispiel einer Job Shop Fertigung

kassel university press

Die vorliegende Arbeit wurde vom Fachbereich Maschinenbau der Universität Kassel als Dissertation zur Erlangung des akademischen Grades eines Doktors der Ingenieurwissenschaften (Dr.-Ing.) angenommen.

Erster Gutachter: Prof. Dipl.-Ing. A. Reinhardt
Zweiter Gutachter: Prof. Dr.-Ing. U. Bracht

Tag der mündlichen Prüfung 27. Februar 2007

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.ddb.de abrufbar

Zugl.: Kassel, Univ., Diss. 2007
ISBN 978-3-89958-289-5
URN: urn:nbn:de:0002-2897

www.upress.uni-kassel.de

Druck und Verarbeitung: Unidruckerei der Universität Kassel
Printed in Germany

Vorwort

Diese Arbeit entstand während meiner Tätigkeit in der Nutzfahrzeugplanung der DaimlerChrysler AG am Standort Gaggenau, während der Phase der Einführung digitaler Planungsmethoden von 2002 bis 2005. Die wissenschaftliche Betreuung erfolgte durch Herrn Univ.-Prof. Dipl.-Ing. A. Reinhardt, Leiter des Fachgebiets Produktionssysteme des Instituts für Produktionstechnik und Logistik der Universität Kassel. Bei ihm möchte ich mich herzlich für die konstruktiven und interessanten Diskussionen über das Thema, die Wissenschaft im Allgemeinen und die Gesellschaft bedanken. Weiterhin gilt mein besonderer Dank Herrn Prof. Dr.-Ing. U. Bracht für die Übernahme des Koreferats. Meinem Doktorandenkollegen Dipl.-Ing. Nemrude Verzano am Institut von Prof. Reinhardt möchte ich für die gemeinsamen und erfolgreichen Entwicklungsfortschritte und den guten und interessanten Informationsaustausch danken.

Innerhalb der DaimlerChrysler AG möchte ich mich bei den vielen Kolleginnen und Kollegen bedanken, welche im Zusammenhang mit der Einführung der digitalen Planungsmethoden aus allen erdenklichen Fachrichtungen ihr Wissen, ihre Erfahrung, wichtige Ratschläge und auch konstruktive Kritik wohlwollend und bereitwillig zur Verfügung gestellt haben und das auch heute noch tun. Ganz besonders möchte ich mich bei meinem Abteilungsleiter Herrn Achim Schäfer und meinem Teamleiter Herrn Dipl.-Ing. Friedrich Först bedanken, die mir freies Geleit bei der Entwicklung neuer Methoden gegeben haben und mit ihrer Geduld, Erfahrung und nicht zuletzt auch mit finanziellen Mitteln die im Zusammenhang mit dieser Arbeit stehenden Methoden erst ermöglicht haben. Meinem Nachfolger als Doktorand Herrn Dipl.-Wi.-Ing. Ingo Hotz möchte ich vielmals für die schon während seiner Diplomarbeit eingebrachten Ideen, sein Engagement und die Begeisterung für das Thema „Digitale Fabrik“ danken. Ich wünsche ihm bei seinem Promotionsvorhaben viel Erfolg und stets die Muße, das Vorhaben mit Freude und Durchhaltevermögen zu Ende zu führen.

Meiner Familie und meiner Freundin Andrea danke ich für ihr Verständnis, die aufbauenden Worte und die Geduld, die sie während dieser Zeit für mich aufgebracht haben. Außerdem möchte ich mich noch bei Herrn Dipl.-Ing. Hans-Ulrich Wacker bedanken, der mir sehr geholfen hat, das Ziel nie aus den Augen zu verlieren.

The best way to predict the future is to invent it. — Alan Kay

Karlsruhe, im Dezember 2006

Kurzfassung

Die Marktsituation im weltweiten Automobilsektor wird durch die fortschreitende Globalisierung immer straffer. Schon seit den 70er-Jahren wandelt sich der Verkäufermarkt zum Käufermarkt [WKD00, Kmu00]. Die Unternehmen reagieren mit immer kürzeren Produktlebenszyklen, vielen kundengerechten Varianten und einem größeren Produktportfolio. Diese Entwicklung macht sich besonders im Bereich der Produktentstehung und Produktionsplanung bemerkbar. Zur Bewältigung und Absicherung des kontinuierlich wachsenden Planungsumfangs werden heute verstärkt die digitalen Planungsmethoden und insbesondere die zugehörige Materialflusssimulation eingesetzt.

In den interdisziplinären Planungsteams, welche häufig aus Experten der Entwicklung, Planung, Produktionstechnik und Logistik bestehen, ist der Experte für Materialflusssimulation immer öfter ein fester Bestandteil. Seine Aufgabe ist es, die Konzepte und Informationen der Projektbeteiligten in ein aussagekräftiges Simulationsmodell umzuwandeln und anschließend mit der Projektgruppe anhand der Simulationsergebnisse die bestmögliche Alternative auszuwählen. Ein hoher Zeitanteil entfällt dabei auf die Beschaffung und Zusammenführung der von der Materialflusssimulation benötigten Daten. Zur Bewältigung dieser Aufgabe und zur Beschleunigung der Modellerzeugung wird im Rahmen dieser Arbeit ein Verfahren zur teilautomatisierten Modellgenerierung aus bestehenden Planungsdaten entwickelt. Die Daten sind in autonomen, verteilten Datenbanksystemen, welche von der Planung zur Dokumentation genutzt werden, vorhanden und müssen nicht mehr vom Simulationsexperten manuell in ein Modell überführt werden. Damit Modell und Planungszustand immer in einem synchronen Zustand bleiben und Dateninkonsistenzen vermieden werden, beruht das Verfahren auf einer Online-Anbindung dieser Systeme.

Das Verfahren ermöglicht die Erzeugung eines Grundmodells mit vielen simulationsrelevanten Parametern, welches einerseits zur schnellen Ableitung eines Grobplanungsmodells genutzt werden kann und andererseits als Basismodell zur Erzeugung komplexer Modelle dienen soll. Die Methode Materialflusssimulation soll hierdurch besser in den Planungsprozess integrierbar werden und somit die weitere Ausdehnung der heutigen Anwendungsfelder unterstützt werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

3D	Dreidimensional
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
Avo	Arbeitsvorgang
BCL	Batch Control Language
CAD	Computer Aided Design
CAPE	Computer Aided Production Engineering
CIM	Computer Integrated Manufacturing
CIRP	College International pour la Recherche en Productique
CODASYL	Conference on Data Systems Languages
Con	Connection
ConWIP	Constant Work In Process
DB	Datenbank
DBI	Database Independent
DBMS	Datenbank-Management-System
DIN	Deutsche Industrienorm
DLL	Dynamic Link Library
DMU	Digital Mockup
DOM	Document Object Model
DTD	Document Type Definition
DV	Datenverarbeitung
DXF	Drawing exchange format
EAI	Enterprise Application Integration
ebXML	Electronic Business XML
EDI	Electronic Data Interchange
EDV	Elektronische Datenverarbeitung
ERP	Enterprise Resource Planning
FIFO	First In First Out
FTP	File Transfer Protocol
FTS	Fahrerlose Transportsysteme
GI	Gesellschaft für Informatik
GPSS	General Purpose Simulation System

GUI	Graphical user interface
HLA	High Level Architecture
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IAT	Interarrivaltime
IBM	International Business Machines
IGES	Initial Graphics Exchange Specification
IP	Internet Protocol
ISO	International Organization for Standardization
IT	Informationstechnik
ITE	Innerbetriebliche Transporteinheit
JAXB	Java Architecture for XML Binding
JDBC	Java Database Connectivity
JIS	Just in Sequence
JIT	Just in Time
JNDI	Java Naming and Directory Interface
JVM	Java Virtual Machine
LAN	Local Area Network
LKW	Lastkraftwagen
MRP	Material Requirements Planning
MRP II	Manufacturing Resources Planning
MS	Mehrmaschinenfaktor
MTBF	Meantime between failiure
MTM	Methods Time Measurement
MTTR	Meantime to repair
NC	Numeric Control
ODBC	Open Database Connectivity
OMF	(Wheater) Observation Markup Format
oo	objektorientiert
OPC	OLE for Process Control
PDF	Portable Document Format
PKW	Personenkraftwagen
PPD	Produktions-, Planungs- und Dokumentationssystem
PPR	Produkt-Prozess-Ressource
PPS	Produktionsplanung und -steuerung
REFA	Verband für Arbeitsgestaltung, Betriebsorganisation und Unternehmensentwicklung e.V.
rel.	relational
SAP	Systeme, Anwendungen und Produkte in der Datenverarbeitung AG

SAX ..... Simple API for XML processing
SCL ..... Simulation Control Language (Quest)
SDX ..... Simulation Data Exchange
SE ..... Simultaneous Engineering
SGML ..... Standard Generalized Markup Language
SOA ..... Service Oriented Architecture
SOP ..... Start of Production
SPC ..... Statistical Process Control
SQL ..... Structured Query Language
STEP ..... Standard for the Exchange of Product model data
TP ..... Transaction Processing
UGS ..... Unigraphics Solutions Corp.
UNIX ..... Uniplexed Information and Computing System
VDBMS ..... Verteiltes Datenbankmanagementsystem
VDBS ..... Verteiltes Datenbanksystem
VDI ..... Verein Deutscher Ingenieure
W3C ..... World Wide Web Consortium
WAN ..... Wide Area Network
WWW ..... World Wide Web
XML ..... Extensible Markup Language
XPath ..... XML Path Language

Kapitel 1

Einleitung

1.1 Zielsetzung

Simulation in Produktion und Logistik wird heute häufig eingesetzt. Kürzere Produktlebenszyklen, höhere Variantenvielfalt und verstärkte Konkurrenz am Markt führen zu der Notwendigkeit höherer Planungssicherheit, kürzerer Planungszeiträume und besserer Datenintegrität. Hierbei erweist sich die Materialflusssimulation als nützliches und leistungsstarkes Hilfsmittel.

Der Aufwand zur Erstellung eines validen Simulationsmodells mit hohem Detaillierungsgrad und dadurch starker Aussagekraft ist immens und erfordert umfangreiches Expertenwissen sowohl im IT- als auch im technischen Bereich (vgl. Abb. 1.1). In der Vergangenheit hatte das zur Folge, dass sich die Simulation nicht direkt in den Planungsprozess integrieren ließ. Dabei kann der volle Nutzen aus der Simulation erst dann gezogen werden, wenn diese durchgängig im Produktentstehungsprozess angewendet wird.

Abb. 1.1 zeigt die prozentuale Verteilung der Zeitaufwände bei der Durchführung einer Simulationsstudie. Je nach Art und Zielsetzung der Anwendung variieren diese. Betrachtet man lediglich den Median in der Darstellung, so entfallen auf die Datenerhebung 28% und auf die Modellerstellung 8% des Gesamtaufwands. In Summe machen diese beiden Schritte also mehr als ein Drittel der Gesamtdauer aus. Betrachtet man nun die Informationssysteme in den Unternehmen, so stellt man fest, dass sehr viele der Daten in elektronischer Form über mehrere Systeme verteilt vorliegen. Der Simulationsexperte muss diese mühsam zusammensuchen oder in Gesprächen mit Planern, Meistern, Instandhaltern, etc. sein Grundverständnis zur Erstellung eines Modells erlangen (vgl. Abb. 1.2).

Genau an dieser Stelle soll die vorliegende Arbeit ansetzen. Da schon ein großer Teil der benötigten Daten in elektronischer Form vorliegt, muss eine Möglichkeit geschaffen werden, diese in ein für die Simulation verwertbares Format zu bringen. Bevor jedoch eine Zusammenführung vorgenommen werden kann, muss vorerst ein Standard zur Modellbeschreibung geschaffen werden, auf den die vorhandenen Daten abgebildet werden können. Dieser soll von unterschiedlichen Simulatoren lesbar sein. Als Vorlage dienen hierbei bekannte Formate aus dem CAD (z.B. DXF, STEP) und das Simulation Data Exchange (SDX) Format. Liegen die Daten in einer einheitlichen Form vor, so ist es naheliegend, diese auch rechnergestützt im Simulationssystem in Modellkomponenten zu überführen. Es entsteht folglich ein Grundgerüst eines Modells, das vom Simulationsexperten den Ansprüchen entsprechend angepasst werden kann.

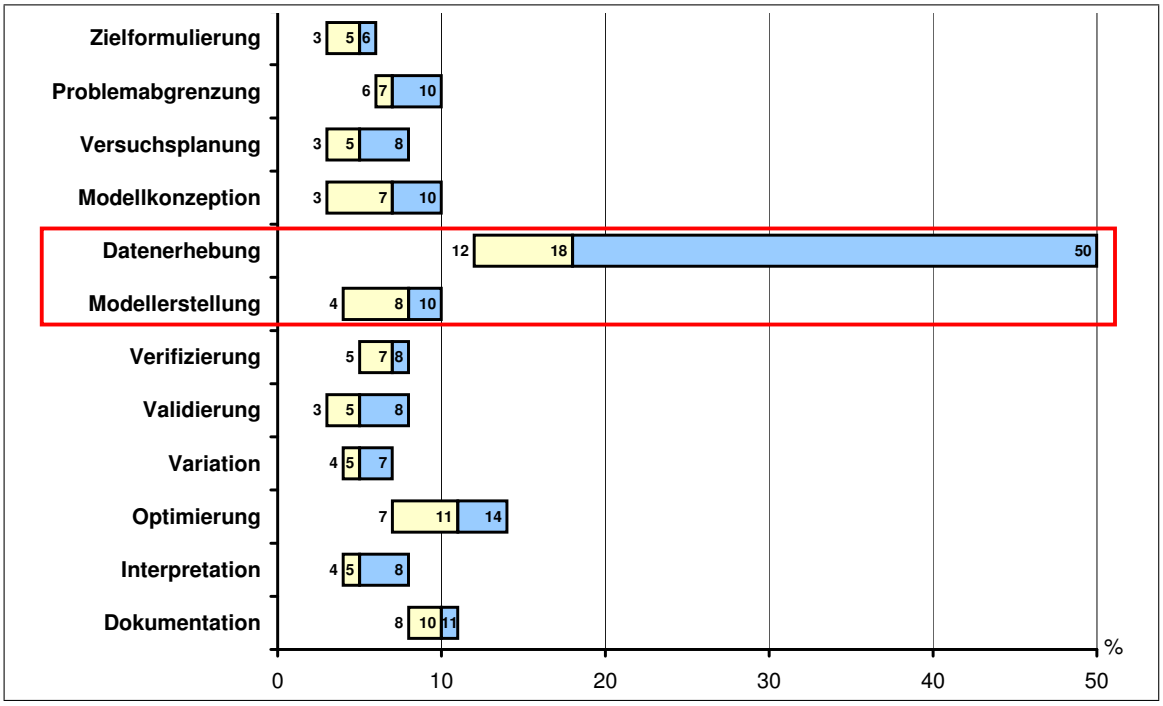


Abb. 1.1: Zeitliche Aufwandsverteilung beim Simulationseinsatz nach [Acè92]

Im Bereich der Datenzusammenführung sollen Webtechniken eingesetzt werden, wie sie aus dem Internet bekannt sind (z.B. Online-Banking, Flugbuchungen, etc.). Diese haben in der Praxis eine hohe Leistungsfähigkeit bewiesen und sind aus dem alltäglichen Leben nicht mehr wegzudenken. Zur standardisierten Beschreibung von Simulationsmodellen muss eine formale Beschreibungssprache eingesetzt werden. Hierfür soll XML[1] genutzt werden, weil es sich sehr gut zur strukturierten Beschreibung von Daten eignet. Außerdem existieren sehr viele webbasierte Techniken zur Umwandlung und Verarbeitung dieses Formats.

Nach dem Praxiseinsatz der im Rahmen dieser Arbeit entwickelten und umgesetzten Methodik haben sich schnell weitere Anwendungsmöglichkeiten in einem Industriebetrieb gezeigt. Die Simulation benötigt sehr umfangreiches und genaues Datenmaterial zur Abbildung der Realität. Diese Daten sind jedoch nicht nur simulationsrelevant, sondern auch für viele weitere Anwendungen im Bereich der Planung und Produktion notwendig. Daher sollen auch diese Anforderungen berücksichtigt werden.

1.2 Motivation

In der Getriebeproduktion am Standort Gaggenau der DaimlerChrysler AG ist die Simulationsunterstützung in der Vergangenheit nur für einzelne Teilbereiche in der Produktion geleistet worden. Der hohe Erstellungsaufwand und die meist fehlende Wiederverwendbarkeit der Modelle hat zur Folge gehabt, dass sich die Simulation nicht in den Planungsalltag integrieren ließ. Hinzu

[1] **E**xtensible **M**arkup **L**anguage – Eine plattformunabhängige Methode zum Strukturieren von Informationen

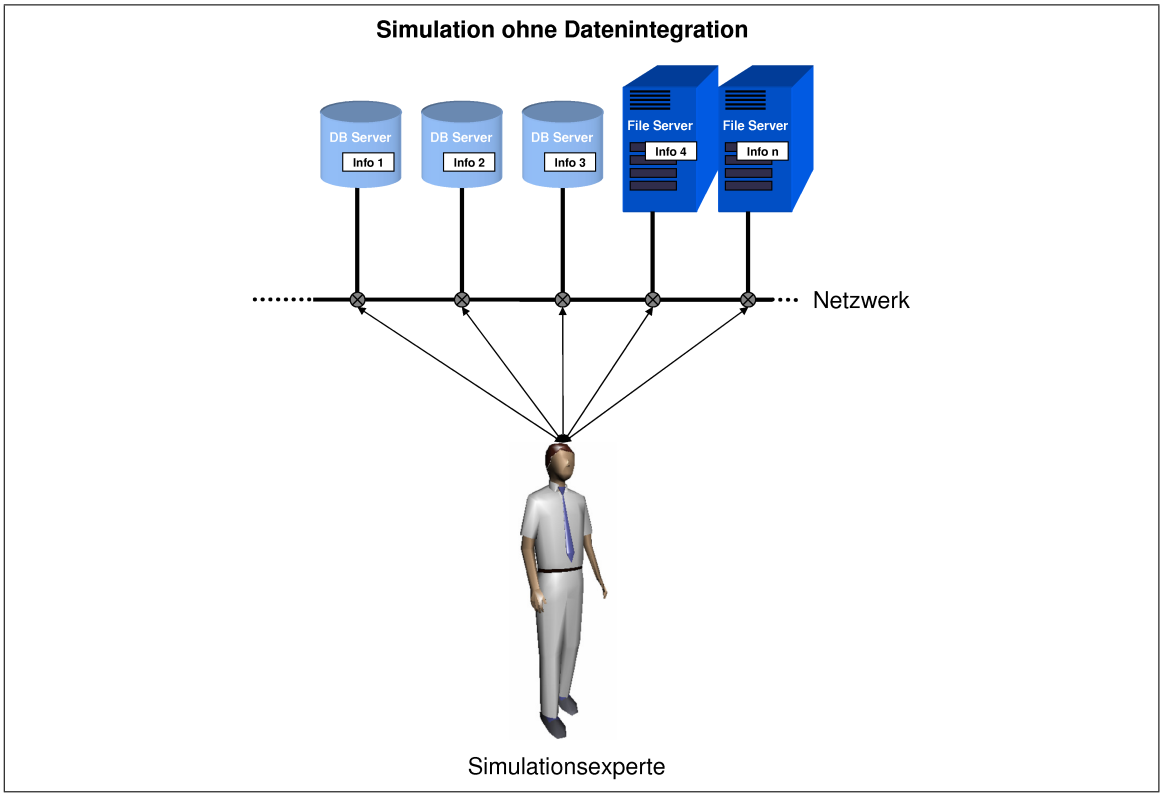


Abb. 1.2: Ausgangssituation dieser Arbeit

kommt, dass in der Vergangenheit Simulationsmodelle nicht direkt auf Daten aus DV-Systemen zugreifen konnten und somit mangels weiterführender Pflege ihre Aktualität verloren haben.

Zu Beginn dieser Arbeit stand deshalb der Gedanke eine Möglichkeit zu finden, Simulationssysteme möglichst effizient in die Planung integrieren zu können und dadurch die teilweise fehlende Akzeptanz unter den Produktionsplanern abzubauen. Vorerst ist eine Analyse des bereits bestehenden Datenmaterials durchgeführt worden, welche gezeigt hat, dass ein Großteil der benötigten Daten für Simulationsstudien bereits in verschiedenen Datenbanksystemen am Standort vorhanden ist. Auch in anderen Unternehmen, welche Simulation aktiv anwenden, wird bestehendes Datenmaterial nur unzureichend oder gar nicht für Simulationsstudien genutzt. Deshalb soll im Rahmen dieser Arbeit der Versuch unternommen werden, die technischen Hintergründe näher zu beleuchten und somit die Lücke zwischen Planungs-, Dokumentations- und Simulationssystemen zu schließen. Dabei liegt der Nutzen im Zeitgewinn bei der Datensammlung, Modellierung und Prüfung der Datenintegrität. Diese Vorteile ermöglichen einen integrierten Einsatz der Methode Materialflusssimulation im Planungsprozess.

1.3 Aufbau der Arbeit

In Kapitel 2 werden die Entstehung und Anwendungsgebiete der Materialflusssimulation insbesondere im Bereich der Automobilindustrie beschrieben. Im Vordergrund steht der aktuelle

Stand der Technik und die Anwendung in der Teilefertigung, anhand derer die Methodik eingesetzt werden soll.

Kapitel 3 gibt eine Einführung in verteilte Datenbanksysteme unter dem Aspekt kooperierender Datenbanken in Unternehmen. Dabei werden die Schwierigkeiten im Zusammenhang mit historisch gewachsenen IT-Systemen erläutert, die Möglichkeiten zum Datenaustausch genannt und eine Integration von Simulationssystemen in diese Datenwelt beschrieben.

Darauf aufbauend wird in Kapitel 4 ein Konzept zur Einbindung autonomer, verteilter Datenbanksysteme in den Datenkreislauf von Simulationsanwendungen entwickelt. Außerdem wird eine wissenschaftliche Abgrenzung der Arbeit zu bereits vorhandenen Integrationsansätzen durchgeführt.

Ein Ansatz zur Abstrahierung der realen Fertigung, um teilautomatisiert Simulationsmodelle generieren zu können wird in Kapitel 5 aufgezeigt.

Das in Kapitel 4 dargestellte Konzept zur Datenzusammenführung und die in Kapitel 5 geschaffenen Rahmenbedingungen seitens der Simulationssysteme werden in Kapitel 6 programmiertechnisch umgesetzt. Dabei werden die einzelnen Schritte aufgezeigt und die eingesetzten Techniken näher beschrieben. Hierbei kommen webbasierte Techniken aus der Client-/Server-Anwendung zum Einsatz, die heute den Stand der Technik in der vernetzten Welt bilden.

Kapitel 7 beinhaltet Erfahrungen aus dem Praxiseinsatz des Umgesetzten Konzepts und zeigt technische Möglichkeiten und Grenzen auf. Es werden weitere Einsatzgebiete des Verfahrens und Möglichkeiten zur Weiterentwicklung dargestellt.

Eine Zusammenfassung findet sich in Kapitel 8.

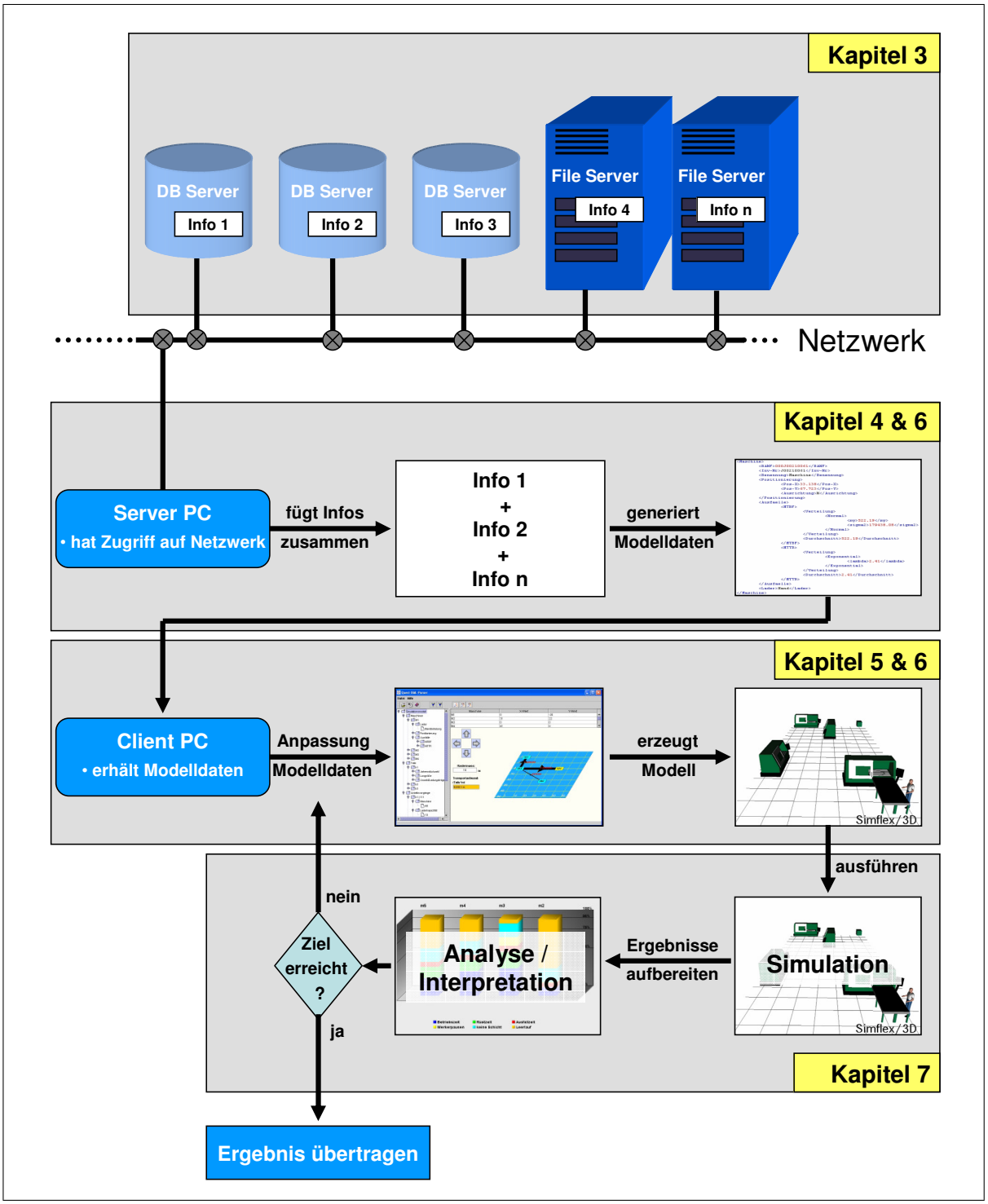


Abb. 1.3: Aufbau der Arbeit

Kapitel 2

Simulation in der Automobilindustrie

> Wir können nur eine kurze Distanz in die Zukunft blicken, aber dort können wir eine Menge sehen, was getan werden muss.
>
> *(Alan Turing)*

Der Begriff Simulation leitet sich von dem lateinischen Wort *simulare* ab und bedeutet sinngemäß „nachbilden“, beziehungsweise „vortäuschen“. Nachgebildet wird ein Szenario immer dann, wenn das Ausführen in der Realität aus verschiedensten Gründen zu vermeiden ist. Ursache und Auslöser ist dabei stets eine Fragestellung, die es zu beantworten gilt, welche jedoch in der realen Welt aus Komplexitäts- oder Kostengründen nicht mit vertretbarem Aufwand zu bewältigen ist [DD02]. So vielseitig das Verb „nachbilden“ im Sprachgebrauch anwendbar ist, so weitreichend ist auch das Einsatzgebiet der Simulation im technischen Bereich. Es liegt in der Natur des Menschen, Dinge zu hinterfragen und nach Antworten zu suchen. Deshalb wird in beinahe allen Disziplinen der Wissenschaft und in der Praxis, wie beispielsweise in der Physik, im Maschinenbau, und der Chemie Simulation angewendet, um komplexe Fragestellungen zu beantworten [Bac96].

2.1 Entstehung, Definition und Einsatzgebiete

Die Geschichte der Computersimulation geht zurück bis in die 50er Jahre, als unter militärischer Geheimhaltung erste Schritte unternommen wurden, mit Hilfe der Simulation die Entwicklung von Kernwaffen während des Manhatten Projekts zu verbessern und auch zu beschleunigen [Wik07a]. Das erste universelle Simulationssystem wurde 1960 von Jeoffrey Gorden (IBM) entwickelt und unter dem Namen GPSS [1] bekannt [Gor61]. Bis heute sind die Anwendungsmöglichkeiten und vor allem auch die Fähigkeiten dieser Systeme in gleichem Maße gewachsen, wie die Technologie und Geschwindigkeit der Rechner, derer sie sich bedienen. Die Entwicklungsgeschwindigkeit im Hardwaresektor ist bis heute noch zum Teil ausschlaggebend für die Möglichkeiten, die durch die digitalen Planungsmethoden entstanden sind. Sehr schnell kam in der Vergangenheit die Forderung nach grafischer Nachvollziehbarkeit bei der Abbildung realer Prozesse im Rechner, denn es liegt auch in der Natur des Menschen nur das zu glauben, was er sieht [Rei77]. Es ist jedoch ein Irrglaube zu denken, dass sich die Simulation von Beginn an stets großer Beliebtheit erfreute und als Methode für die Zukunft gefeiert wurde. Die erste

[1] General Purpose Simulation System

Vorstellung des grafisch-interaktiven Simulators SIMFLEX auf einer Tagung der Gesellschaft für Informatik (GI) 1977 in München endete als Lacherfolg für das Publikum und der Begriff des „Mäusekinos“ war geprägt [Rei03].

Einigen Pionieren in der Vergangenheit, welche ihren Glauben in die Möglichkeiten und den Nutzen der Rechnersimulation nie verloren haben, ist es zu verdanken, dass die heutige Generation von Nachwuchsingenieuren auf sehr leistungsfähige, wenngleich auch nicht besonders anwenderfreundliche, Simulationssysteme zurückgreifen können. Dabei sind den Einsatzfeldern technisch fast keine Grenzen mehr gesetzt. Die Einsatzgebiete der Simulationstechnik lassen sich nach Koŝturiak in fünf wichtige Anwendungsbereiche unterteilen [KG95]:

- Entscheidungsunterstützung (z.B. das Aufzeigen der Folgen bestimmter Entscheidungsmöglichkeiten),
- Planung (z.B. Alternativenvergleich bei der Prozessneugestaltung),
- Forschung und Entwicklung (z.B. Simulation chemischer Reaktionen),
- Organisationsgestaltung (z.B. Bewertung neuer organisatorischer Maßnahmen),
- Ausbildung und Training (z.B. Fahr- und Flugsimulationen).

Unter Simulation versteht man im Allgemeinen und übergeordneten Sinn

> *„die modellhafte Darstellung oder Nachbildung bestimmter Aspekte eines vorhandenen oder zu entwickelnden [..] Systems [..], insbesondere auch seines Zeitverhaltens“.* — [Bro02]

Ein für die Simulation verwendetes Modell, als Abbild der Realität, ist in der VDI-Richtlinie 3633 definiert als

> *„[..] eine vereinfachende Nachbildung eines geplanten oder real existierenden Originalsystems und -prozesses in einem anderen begrifflichen oder gegenständlichen System. Es (Anm.: das Modell) unterscheidet sich hinsichtlich der untersuchungsrelevanten Eigenschaften nur innerhalb eines vom Untersuchungsziel abhängigen Toleranzrahmens vom Vorbild“.* — [VDI05]

Häufig lassen sich so gebildete, wenig umfangreiche Modelle mit analytischen Methoden darstellen, die zu exakten Aussagen über das Verhalten eines Systems führen. Viele der praxisrelevanten Systeme sind jedoch aufgrund der Komponentenzahl und verschiedenen Abhängigkeiten so komplex, dass sich jene nicht oder nur schwer analytisch beschreiben lassen. Die rasante Entwicklung in der Computertechnologie bietet heute die Möglichkeit, komplexe Modelle im Rechner nachzubilden, um mit veränderten Randbedingungen die Abläufe im System zu analysieren. Nach VDI-Richtlinie 3633 versteht man unter Simulation

> *„das Nachbilden eines dynamischen Prozesses in einem System mit Hilfe eines experimentierfähigen Modells, um zu Erkenntnissen zu gelangen, die auf die Wirklichkeit übertragbar sind“.* — [VDI05]

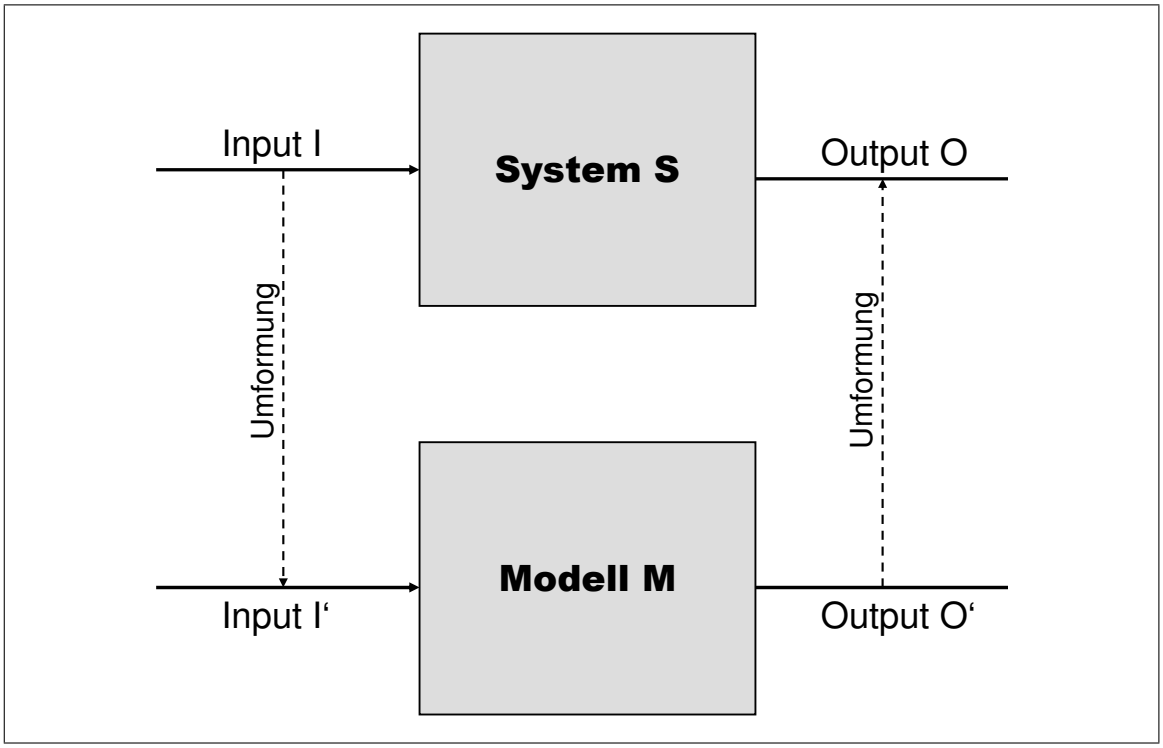


Abb. 2.1: Zusammenhang System - Modell [OS99]

Der Begriff System bezieht sich dabei auf eine abgegrenzte Anordnung von Komponenten, die untereinander durch Abhängigkeiten und Wechselwirkungen in Beziehung stehen. Es ist gekennzeichnet durch:

- Eine Systemgrenze, über die Materie, Energie und Informationen ausgetauscht werden.
- Eine Anzahl von Komponenten, die aus Subsystemen oder nicht weiter zerlegbaren Elementen bestehen.
- Eine festgelegte Ablaufstruktur in den Komponenten.
- Eine Aufbaustruktur, die eine Verknüpfung der Komponenten untereinander festlegt.
- Die definierten Zustände der einzelnen Komponenten und die Art der Zustandsübergänge.

Nach den bisherigen Definitionen sind Modelle nicht an bestimmte Medien gebunden, auf denen sie ausführbar sind. Gehen wir beispielsweise von einem Windkanal aus, so kann man dort mit einem maßstabsgetreuen Modell bestimmte Aerodynamikeigenschaften simulieren. Mit einem Flugzeug in einer Parabelflugbahn kann man den Zustand der Schwerelosigkeit nachbilden. Im weiteren Verlauf dieser Arbeit soll jedoch der Schwerpunkt auf die Simulation in Produktion und Logistik gelegt werden. Sie bedient sich stets eines rechnerkonformen Modells. Es ist definiert als ein formales, durch entsprechende Simulationssoftware ausführbares Modell. Man sagt auch, ein Modell M simuliert ein System S, wenn durch Umformung des Inputs I des Systems in den Input I' des Simulationsmodells und durch Umformung des zu I' produzierten Outputs O', Output O des interessierenden Systems erhalten werden kann [OS99] (siehe Abb. 2.1).

Besondere Beachtung findet die Simulation immer dann, wenn eine oder mehrere der folgenden Rahmenbedingungen, beziehungsweise Beschränkungen auftreten [DD02]:

- Ein vollständiges mathematisches Optimierungsmodell ist nicht verfügbar und auch nicht mit vertretbaren Kosten entwickelbar.
- Verfügbare analytische Methoden machen vereinfachende Annahmen erforderlich, die den Kern des eigentlich vorliegenden Problems verfälschen.
- Ein praktikabler Einsatz analytischer Methoden ist zu kompliziert, beziehungsweise mit erheblichem Aufwand verbunden.
- Es ist zu komplex, kostspielig oder riskant, reale Experimente durchzuführen.

2.1.1 Simulationsarten

Auch nach der Einschränkung der Anwendungsfelder auf den Bereich der Simulation in Produktion und Logistik gibt es noch diverse Ausprägungen, nach denen sich diese unterscheiden lassen. Eine übergeordnete Einteilung kann aufgrund der verwendeten Daten getroffen werden. Sind sämtliche Größen deterministisch, oder lassen sie sich ohne Verlust an Informationen als solche behandeln, so spricht man von deterministischer Simulation. Ist mindestens eine der verwendeten Größen zufallsabhängig, verwendet man den Begriff der stochastischen Simulation.

Eine weitere Untergliederung kann aufgrund der verwendeten Zeitsteuerung vorgenommen werden. Generell zu unterscheiden sind Modelle ohne Zeitbezug, zeitkontinuierliche und zeitdiskrete Modelle. Bei einem Modell ohne Zeitbezug treten keine zeitabhängigen Zustandsänderungen auf. Die inhärente Dynamik der zeitdiskreten und zeitkontinuierlichen Modelle existiert bei ihnen folglich nicht. Man nennt diese Art von Modellen auch statische Modelle [Pag91]. Im Gegensatz hierzu ändert sich der Zustand des Modells, repräsentiert durch die Zustände der einzelnen Objekte, bei der zeitkontinuierlichen und der zeitdiskreten Simulation mit dem Fortschreiten der Zeit. Hierbei entspricht die Simulationszeit der im realen System voranschreitenden Zeit, sie steht jedoch in keinerlei Bezug zur Rechenzeit bei der Ausführung des Simulationsmodells. Die Art und Weise, wie die Simulationszeit voranschreitet und die Durchführung der damit verbundenen Zustandsänderungen des Modells können auf unterschiedlichem Wege erfolgen und sind charakteristisch für die verschiedenen Simulationsmethoden. Die Zeitfortschreibung kann sowohl stochastisch als auch deterministisch erfolgen [DD02].

Ein Repräsentant für die stochastische Simulation ist die Monte-Carlo-Simulation, bei der man von zwei grundlegenden Eigenschaften ausgeht, die typischerweise auch für das Roulett zutreffen: Die Wahrscheinlichkeiten sind für alle Ereignisse bekannt und gleich groß. Des Weiteren sind alle Ereignisse voneinander unabhängig. Diese Art der Simulation eignet sich besonders zur Analyse statistischer Fragestellungen mit bekannten Wahrscheinlichkeitsverteilungen [DD02],[Bla00].

Die kontinuierliche Simulation findet ihre Anwendung in der Analyse von Systemen, bei denen Zustandsvariablen aufgabenbezogen nur kontinuierlich darstellbar sind. Dabei kommen mathematische Modelle zum Einsatz, die zumeist in Form von Differential- bzw Differenzengleichungssystemen vorliegen und dadurch den Zusammenhang zwischen Zeitfortschritt und der Änderung der Zustandsvariablen beschreiben [FPW90]. Man muss jedoch beachten, dass diese Art der Simulation eigentlich nur auf Analogrechnern direkt ausgeführt werden kann. Digitale Rechner können kontinuierliche Modelle nur annäherungsweise berechnen, da sie, bedingt durch ihren Aufbau, nur diskrete Werte verarbeiten können. Abhilfe schaffen hierbei Näherungsverfahren, wie beispielsweise nach Newton oder Riemann [DP88]. Aufgrund dieser Tatsache spricht man in diesem Fall auch von „quasi-kontinuierlicher“ Simulation. Typische Anwendungsgebiete sind die Betrachtung von Brems- und Beschleunigungsvorgängen, Flug- und Crashtestsimulationen.

Die diskrete Simulation befasst sich mit der Modellierung von dynamischen Systemen [DD02]. Der Zustand eines Systems wird durch zeitabhängige Zustandsvariablen beschrieben. Änderungen des Systemzustands erfolgen ausschließlich an bestimmten, diskreten Zeitpunkten. Das Verhalten des Modells zwischen diesen Zeitpunkten wird nicht betrachtet. Die zeitliche Entwicklung eines Systems wird durch eine endliche Folge von Zuständen abgebildet [Fis01]. Bezüglich der Bestimmung der diskreten Zeitpunkte differenziert man die entsprechenden Modelle in zeit- und ereignisgesteuert. Bei der zeitgesteuerten Variante wird die Modellzeit um äquidistante Zeitintervalle Δt erhöht. Nach jeder Erhöhung des Inkrements werden die aufgetretenen Zustandsänderungen innerhalb der letzten Epoche durchgeführt. Man nennt diese Art der Aktualisierung auch synchrone Aktualisierung. Bei infinitesimal kleinem Δt wird das Modell quasi-kontinuierlich. Während die Methodik der fixen Zeitinkremente in der Regel nicht weiter differenziert wird, findet man in der Literatur eine Reihe von methodischen Ansätzen bezüglich der variablen Zeitinkremente [Lie95].
Geprägt sind diese Ansätze jeweils durch ihre sogenannte „Weltsicht“ [Hoo86]. Man versteht darunter auch die Art und Weise, wie Ereignisse und Zustände der realen Welt im Modell repräsentiert werden können. In der Literatur finden sich unterschiedliche Ansichten, bzgl. der Simulationsarten, jedoch findet sich häufig die Untergliederung in vier Klassifikationen:

- Ereignisorientierte Simulation,
- aktivitätsorientierte Simulation,
- prozessorientierte Simulation und
- transaktionsorientierte Simulation.

Diese Ansätze werden bei diversen Autoren ([Meh94, Lie95, Pag91, Noc90]) sehr detailliert beschrieben (vgl. auch Abb. 2.2). Ein Großteil der in Produktion und Logistik verwendeten Simulationssysteme basieren auf dem Konzept der ereignisorientierten Zeitsteuerung. Einer der Gründe hierfür ist die Tatsache, dass die Ablaufsteuerung bei ereignisorientierten Modellen wesentlich einfacher zu realisieren ist, als bei anderen Modellierungsarten [Pag91]. Bei der ereignisgesteuerten Simulation bewirkt das Eintreten eines Ereignisses eine Zeitänderung innerhalb des Modells. Dabei sind Ereignisse Zustandsänderungen des Gesamtsystems, die selbst keine Dauer haben. Zwischen zwei Ereignissen bleibt der Systemzustand per Definition konstant. Es steht

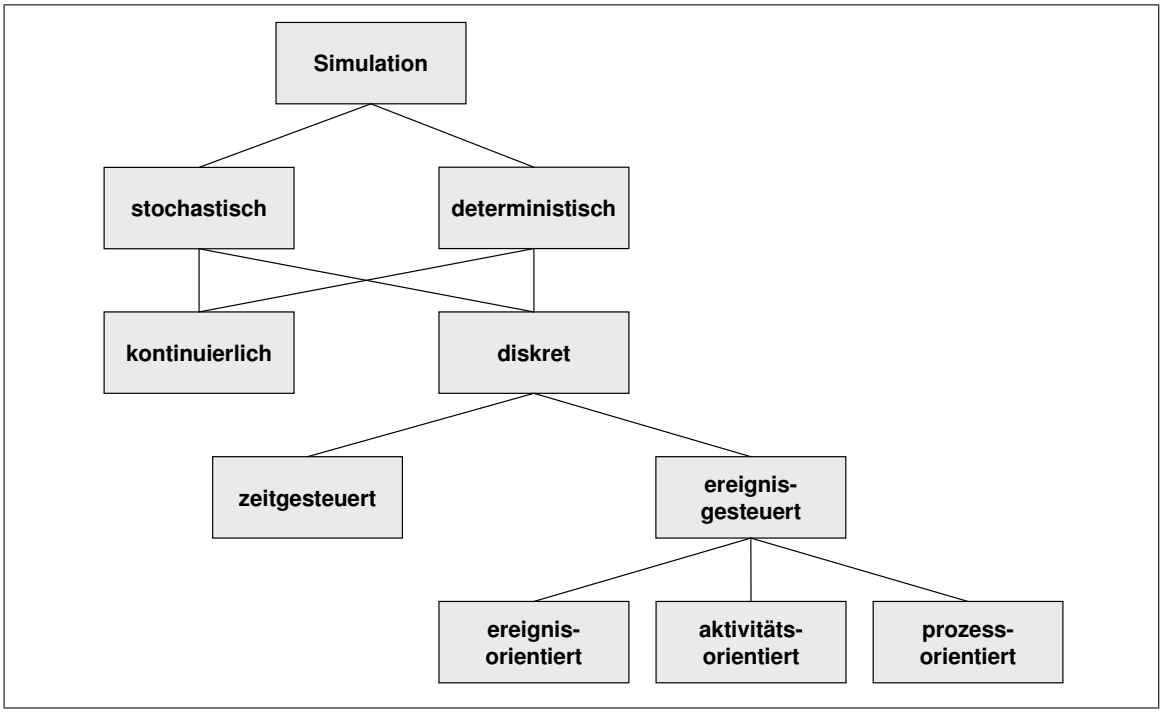


Abb. 2.2: Klassifikation der Simulation (vgl. [Bac96])

somit unmittelbar nach dem Auftreten eines Ereignisses fest, welche weiteren Ereignisse zukünftig auftreten werden. Sie werden vom sogenannten „Scheduler“ in einer temporal sortierten Ereignisliste verwaltet und zur Zeitsteuerung herangezogen. Das Zeitinkrement Δt beschreibt also immer die Zeitspanne bis zum nächsten globalen Ereignis. Vorteil dieser Variante ist eine Einsparung an Rechenzeit, da ereignislose, teilweise länger anhaltende Zeiten ohne Aktivitäten übersprungen werden können. Der Rechner wird lediglich bei der Ausführung der Ereignisaktionen benötigt, um Routinen abzuarbeiten, welche den Systemzustand modifizieren.

2.1.2 Nutzen und Risiken durch Anwendung von Simulation

Der Nutzen und das Potenzial der digitalen Planungsmethoden und der „Digitalen Fabrik“ werden heute insbesondere in der Automobilindustrie hoch gelobt und treten als Lösung vieler Mängel aus der Vergangenheit hervor (vgl. [Aßm02, SS02, Sch00a, Kra01, Wie02, Wor02]). Zu den wesentlichen Vorteilen zählen unter anderem der modulare, teilweise objektorientierte Aufbau von Simulationsmodellen, wodurch diese flexibel an veränderbare Gegebenheiten anpassbar sind. Sie sind wesentlich effizienter anwendbar als analytische Modelle [KG95]. Die Funktionalität eines Systems ist ebenso abbildbar, überprüfbar und bewertbar, wie der Einfluss von stochastischen Größen und die Bedeutung einzelner Systemparameter. Außerdem lassen sich verschiedene Phasen wie Einschwingverhalten, eingeschwungener Zustand, sowie Extremsituationen des Systems gezielt beobachten. Dabei kann mittels Animation das dynamische Systemverhalten verfolgt werden. Der simulierte Prozess kann durch Zeitraffung oder Zeitdehnung so beschleunigt oder auch verlangsamt werden, dass eine visuelle Beobachtung der Veränderun-

gen im System möglich ist. Man muss jedoch stets beachten, dass bei der Umsetzung eines realen Zustands in ein Modell auch die subjektive Auffassung des Modellierers, also des Menschen, einfließt. Dabei ist hinreichend bekannt, dass jeder Mensch eine unterschiedliche Wahrnehmung besitzt, und somit die Ergebnisse einer Simulationsstudie wesentlich von der Interpretation des Anwenders beim Modellieren abhängen [Sch01].

Sobald ein realitätstreues Modell vorliegt, können für jedes beliebige Objekt innerhalb des Systems Kenngrößen und Histogramme evaluiert werden, welche z.B. Aufschluss über Durchlaufzeiten und Kapazitätsauslastungen bringen [KW97]. Ein weiterer Nutzen liegt in der Wiederholbarkeit von Experimenten. Es ist nicht nur möglich bestimmte Kennzahlen während eines Experiments zu ermitteln, sondern aufgrund der Tatsache, dass sich die Simulation beliebig oft mit gleichen oder unterschiedlichen Parametern wiederholen lässt, besteht auch die Möglichkeit statistische Kennzahlen, wie Erwartungswerte und Standardabweichungen, zu ermitteln. Am effizientesten lassen sich digitale Planungsmethoden bei der Planung noch nicht existenter Fabrikanlagen einsetzen. Es können dabei ohne Risiko und bei geringem Kostenaufwand das Verhalten eines geplanten Systems analysiert, Schwachstellen erkannt und anhand der leichten Modifizierbarkeit eines Modells Alternativen miteinander verglichen werden. Simulationsexperimente sind im Planungs- und Projektierungsprozess unmittelbar entscheidungs- und kostenwirksam [Gru99]. Durch die Visualisierung eines Systems im dreidimensionalen Raum wird ein besseres Aufgabenverständnis erzielt und gleichzeitig die Akzeptanz bei den Entscheidungsträgern erhöht. Auch im Bereich der interaktiven Schulung können Simulationsmodelle eingesetzt werden. Das hat den Vorteil, dass Übungen gestaltet werden können, welche das reale System nicht gefährden [Bac96, Sch03].

Die Aussagekraft der Ergebnisse aus einem Experiment hängt sehr stark von der Qualität der Eingangsdaten und der Validität des Modells ab. Die Datenerfassung, Modellbildung und Auswertung kann daher sehr aufwändig sein und erfordert breites Expertenwissen. Mängel bei der Datenerhebung und der Interpretation sind später nur schwer erkennbar und können schnell zu falschen Entscheidungen führen. Daher muss insbesondere bei den Vorbereitungen zur Modellierung sehr präzise vorgegangen werden.

In der Praxis und in Fachzeitschriften sieht man oftmals Modelle ganzer Fabriken innerhalb eines Systems. Dabei sieht man meist nicht, dass der Arbeitsaufwand bei der Modellierung und Auswertung von großen Systemen überproportional mit der Größe wächst. Dies liegt in der Notwendigkeit begründet, dass eine Vielzahl identischer und unabhängiger Experimente durchgeführt werden muss, um zu einem statistisch gesicherten Ergebnis zu kommen. Auch bei deterministischen Eingangsdaten treten in diesem Fall Schwierigkeiten auf. Bei einer Vielzahl an Input-Parametern wird die Anzahl der möglichen Parameterkombinationen extrem groß (bei n Parametern und jeweils m Parameterausprägungen existieren n^m verschiedene Permutationen). Um eine umfassende Analyse eines simulierten Systems zu erhalten, sollte möglichst für jede Permutation ein Simulationslauf durchgeführt werden. Zur Verminderung der Anzahl der Durchläufe kann eine Klassierung der Ausprägungen erfolgen [CT96].

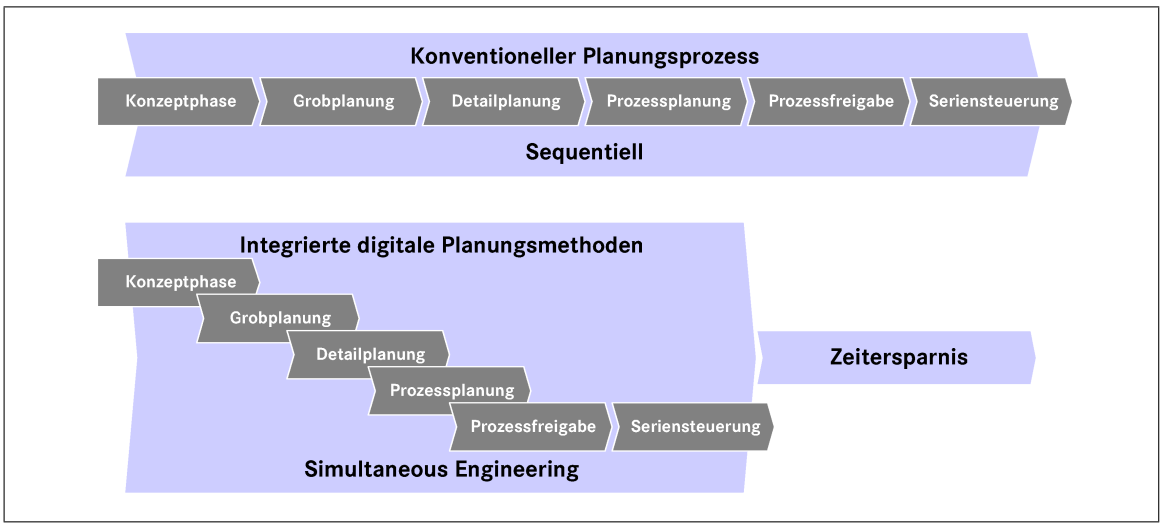


Abb. 2.3: Zeitgewinn durch die Nutzung des Simultaneous Engineering (nach [VB05])

2.2 Stand der Technik in der Materialflusssimulation

Die Marktanforderungen und der Konkurrenzdruck machen es in der Automobilindustrie notwendig nach Methoden zu suchen, welche den Planungsablauf verbessern und die Planungsdauer bei neuen Produktprojekten beschleunigen. Herr Gora von der Adam Opel AG kam zu der Erkenntnis, dass die Besten wissen: „Wer aufhört, besser zu werden, hat aufgehört, gut zu sein!“ [Gor03]. Die Simulation als digitale Planungsmethode ist hierbei eine hilfreiche Unterstützung. Dabei ist die Integration dieser Methode in den Planungsalltag, durch eine in vielen Firmen über die Jahre etablierte Organisation, nicht so einfach wie es scheint. Fragen nach dem effektiven Nutzen, dem kostenmäßigen Mehraufwand und der möglichen Einbettung in den standardisierten Ablauf stellen sich immer wieder aufs Neue. Nachdem heute ein leistungsfähiger Standard an Software zur Bewältigung der Fabriksimulation und auch die notwendige Hardware zu akzeptablen Preisen zur Verfügung steht, entwickelt sich langsam eine neue Disziplin im Berufsbild des Ingenieurs – der „Virtual Reality Ingenieur“. Begründet liegt diese Entwicklung in der Tatsache, dass bei einem effizienten Einsatz digitaler Planungsmethoden verschiedenartige Bereiche innerhalb einer Firma simultan und kooperativ zusammenarbeiten müssen. Außerdem kann der Simulationsanwender Unterstützungs- und Überzeugungsleistung gegenüber dem Management erbringen [Jak91]. Die Forderung nach Simultaneous Engineering (SE) (vgl. Abb. 2.3) im Produktentstehungsprozess wird folglich durch digitale Planungsmethoden unterstützt [Rib00].

Das Zusammenspiel der verschiedenen digitalen Planungsmethoden innerhalb des Produktlebenszyklus wird heute – insbesondere in der Automobilindustrie – oftmals unter dem Dachbegriff „Digitale Fabrik“ zusammengefasst. Die Definition dieses Begriffs hat sich erst in den letzten Jahren mit der wachsenden Bedeutung dessen entwickelt. Eine vorläufige Beschreibung wurde auf der CIRP 2003 präsentiert und umfasste folgende Inhalte:

> *The „Digital Factory“ is the entirety of all methods and tools for the sustainable support of a factory planning and factory operation including the respective processes (workflow) on basis of linked digital models (in connection with the product*

> *model). This definition on the one hand implies the cross-linking of necessary tools and models and on the other hand considering static characteristics and systems dynamic.*
> *The „Digital Factory" does not only require the representation of the physical aspects of the factory including product specifications, logistic and technical processes but also consideration of non-physical aspects such as organizational structures or existing and user know-how.* — [HJW03]

In 2006 wurde innerhalb des VDI-Fachausschusses Digitale Fabrik unter Leitung von Prof. Dr.-Ing. Uwe Bracht von der TU Clausthal die VDI-Richtlinie 4499 geschaffen, welche den Begriff Digitale Fabrik definiert:

> *Die Digitale Fabrik ist der Oberbegriff für ein umfassendes Netzwerk von digitalen Modellen und Methoden unter anderem der Simulation und 3D-Visualisierung. Ihr Zweck ist die ganzheitliche Planung, Realisierung, Steuerung und laufende Verbesserung aller wesentlichen Fabrikprozesse und -ressourcen in Verbindung mit dem Produkt.* — [VDI05]

Bereits in den 80er-Jahren entstand die Vision des Computer Integrated Manufacturing (CIM). Darunter versteht man den „integrierten EDV-Einsatz in allen mit der Produktion zusammenhängenden Betriebsbereichen" [Bac96]. Problematisch bei diesem Ansatz ist jedoch das Verhältnis zwischen Nutzen und Aufwand, da es sich um eine hoch komplexe Methodik handelt, bei der viele verschiedenen Einflussfaktoren und Aspekte berücksichtigt werden müssen [WBN96]. Dennoch besteht die Notwendigkeit für den integrierten EDV-Einsatz in der Produktion. Das Konzept der Digitalen Fabrik ist ein neuer Ansatz zur EDV-Unterstützung der Produktions- und Planungsbereiche. Die Vision der Digitalen Fabrik besteht darin, die wesentlichen technischen Gegebenheiten eines Unternehmens digital darzustellen und die Daten in einem virtuellen Abbild miteinander zu verschmelzen. Mit der daraus entstehenden digitalen Umgebung kann sehr effizient simuliert, analysiert und getestet werden, ohne die Produkte respektive die Produktionsstätten tatsächlich jemals gebaut oder umgebaut zu haben. Typische Fragestellungen, die mit den Methoden der Digitalen Fabrik beantwortet werden können, sind auszugsweise:

- Passt Produkt A in Fertigungszelle X?
- Wie groß müssen Puffer dimensioniert sein?
- Passt Teil B in den Bauraum Y?
- Wo entstehen Engpässe und Überschüsse?
- Was kosteten verschiedene Fertigungsstrategien?

Das Ziel bei der DaimlerChrysler AG besteht bildlich gesprochen darin, dass „zukünftig jedes Digitale Fahrzeug die Digitale Fabrik erfolgreich – das heißt unter Erfüllung der vorgegebenen Kosten-, Qualitäts- und Terminziele – passiert hat" [SS02]. Rechnergestützte Simulations- und Fabrikplanungssysteme sind nun bereits seit den 80er-Jahren verfügbar und doch kann man insbesondere in den vergangenen Jahren einen deutlichen Schub bei der Integration dieser Systeme in der Industrie erkennen. Ein wesentlicher Grund hierfür ist der Wandel in der Wettbewerbssituation, mit dem sich die Industriebetriebe konfrontiert sehen. Die Automobilindustrie hat eine

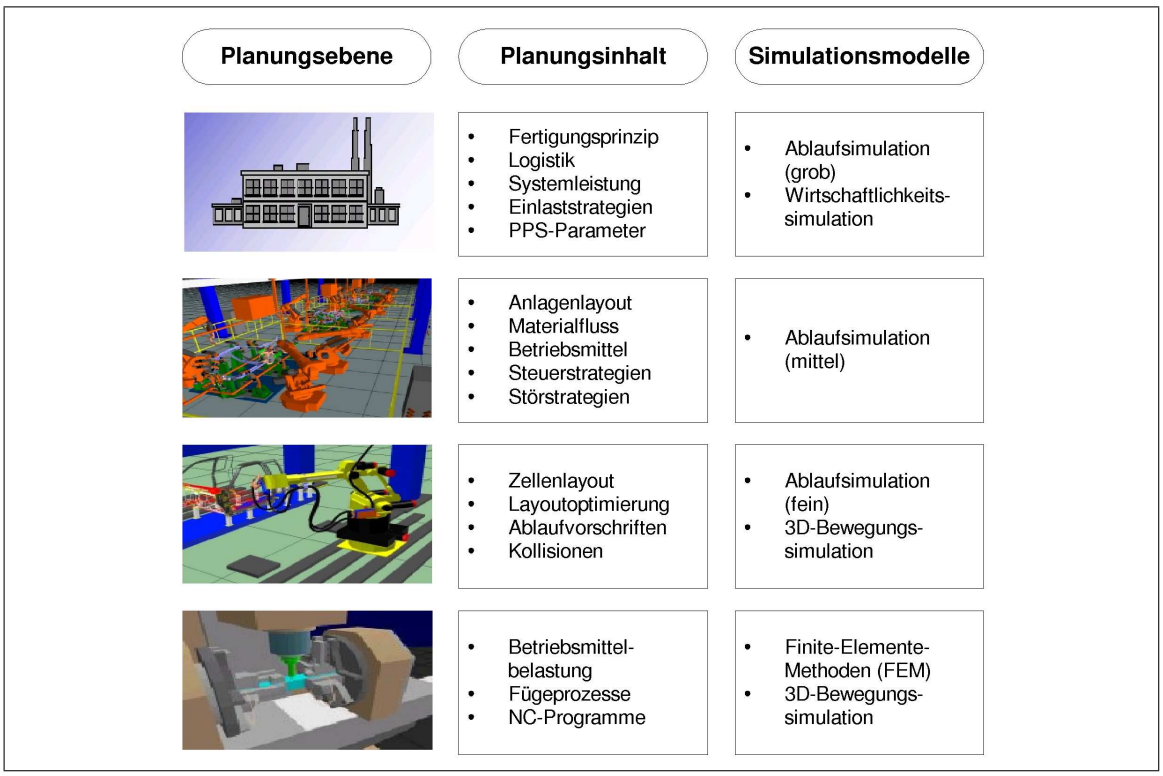


Abb. 2.4: Einsatz von Simulation auf unterschiedlichen Hierarchieebenen (nach [Ama94])

Zeit hinter sich, in der das Endprodukt durch Produktions- und Werkskapazitäten bestimmt wurde. Heute ist nahezu aller Bedarf beim Konsumenten gedeckt und das Wachstum ist gering. Der Verkäufermarkt hat sich zu einem Käufermarkt gewandelt [Fre04].

Die Bedürfnisse der Kunden rücken immer mehr in den Mittelpunkt. Dabei müssen sich die Unternehmen besonders auf eine kundenorientierte Leistungserbringung mit zeitgerechten, qualitativ hochwertigen und kostengünstigen Produkten konzentrieren. Wettbewerb besteht heute aus mehreren Facetten. Neben den Wettbewerb mit neuen Produkten tritt der Wettbewerb der Zeit, der Kosten und der Qualität. Als eine wesentliche Veränderung ist vor allem der Übergang vom Mengenwachstum zum Variantenwachstum zu nennen. Dieses bedingt auch eine zunehmende Innovationsdynamik. Die Zahl und Geschwindigkeit brauchbarer oder auch nur vermeintlicher Innovationen nimmt stetig zu, die Vermarktungsdauer und der Lebenszyklus verkürzen sich. Diese Entwicklung zwingt Unternehmen nicht nur Kosten, sondern auch Zeit zu sparen [Ama94, FR00]. Heute bemüht man sich, diesem Trend mit einem neuen Konzept in der Produktionswirtschaft zu begegnen – der Digitalen Fabrik. Sie erstreckt sich über den ganzen Produktentstehungsprozess, von der Entwicklung über die Grob- und Feinplanung bis hin zur Serienplanung und -steuerung (siehe Abb. 2.4) [VDI02].

Bereits 1999 wurden erste Ansätze für die rechnerintegrierte Planung evaluiert. In kleinen Expertenkreisen, bestehend aus Fachleuten der IT, Produktionsplanung und Entwicklung beschäftige man sich mit der Definition der Anforderungen und der Verifikation des Nutzens. Es wurden Wirtschaftlichkeitsuntersuchungen angestellt, welche den Kosten für eine Einführung neuer Planungsmethoden gegenübergestellt wurden. Erste Piloteinsätze in den Jahren 2001 bis 2003

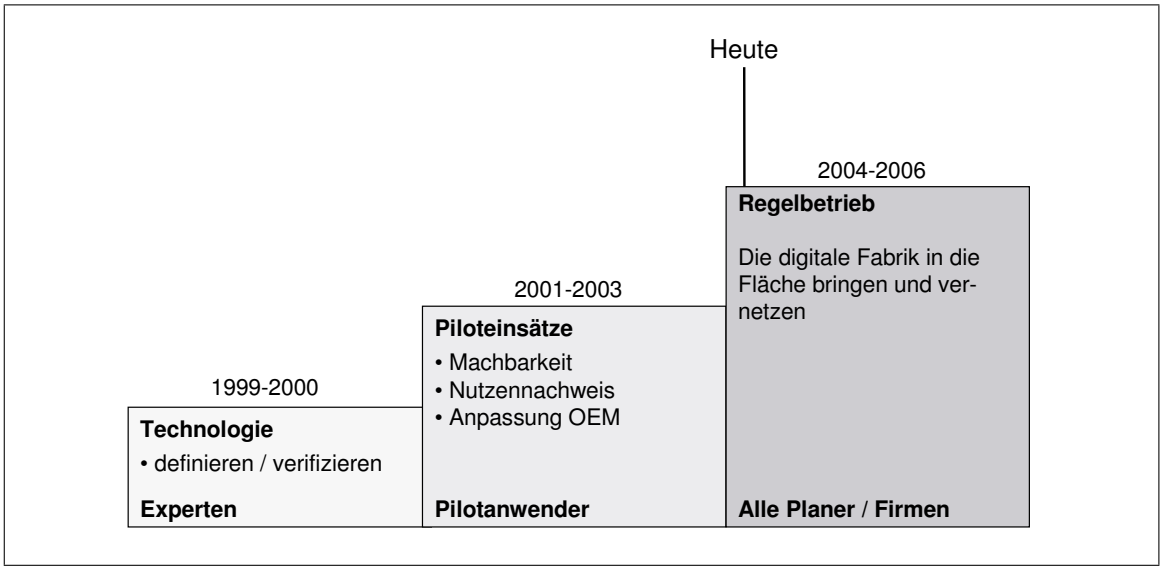


Abb. 2.5: Stand der digitalen Fabrik in der Automobilindustrie [GB05]

haben die prinzipielle Machbarkeit und den Nutzen bestätigt. Jedoch sah man sich auch vielen Herausforderungen gegenübergestellt. Hierzu zählen die Integration der bestehenden Systemlandschaft, die Leistungsfähigkeit der neuen Planungssoftware und auch die menschlichen Hemmschwellen. Nicht jeder Anwender steht neuen Errungenschaften aufgeschlossen gegenüber. Gewohnheiten ändern sich nur sehr langsam. Den überzeugten Pilotanwendern ist es zu verdanken, dass heute der Durchdringungsgrad der digitalen Planungsmethoden stark gewachsen ist. Die Aufgaben rund um dieses Thema sind zwar noch lange nicht alle erledigt, doch wenn es gelingt, die digitale Fabrik kontinuierlich weiter zu integrieren, so wird in den nächsten Jahren ein ganz neuer Standard in der Planungswelt alltäglich (vgl. Abb. 2.5).

2.3 Simulation in der Teilefertigung

Die Simulation in der Automobilindustrie ist heute schon sehr weit fortgeschritten und umfasst, wie bereits dargestellt, die virtuelle Abbildung ganzer Produktionsstätten noch vor der Grundsteinlegung des realen Werkes. Dabei kann man feststellen, dass Simulationsprojekte im Rahmen der Digitalen Fabrik in den meisten Fällen Montagewerke der PKW- und LKW-Industrie abdecken. Betrachtungsschwerpunkte sind dabei die Austaktung von Bandabläufen, die logistischen Rahmenbedingungen (z.B. Materialbereitstellung und Staplerwege) und Baubarkeitsuntersuchungen (z.B. die Planung von Schweißzangen im Rohbau).

Im Bereich der Teilefertigung, zu der u.a. die Aggregatewerke (z.B Motor, Getriebe, Achsen, etc.) gehören, ergeben sich bei Simulationsprojekten sehr diversifizierte Aufgabenstellungen. Die wesentlich höhere Fertigungstiefe, die große Variantenvielfalt und ein wesentlich höherer Einsatz von Fertigungsmaschinen führen zu bestimmten Anforderungen an die Modellierung, die Simulationssysteme und den Ablauf von Simulationsstudien. Zu den wichtigsten Anforderungen gehören die Rüstzeitenminimierung, die Losgrößenoptimierung und eine effiziente

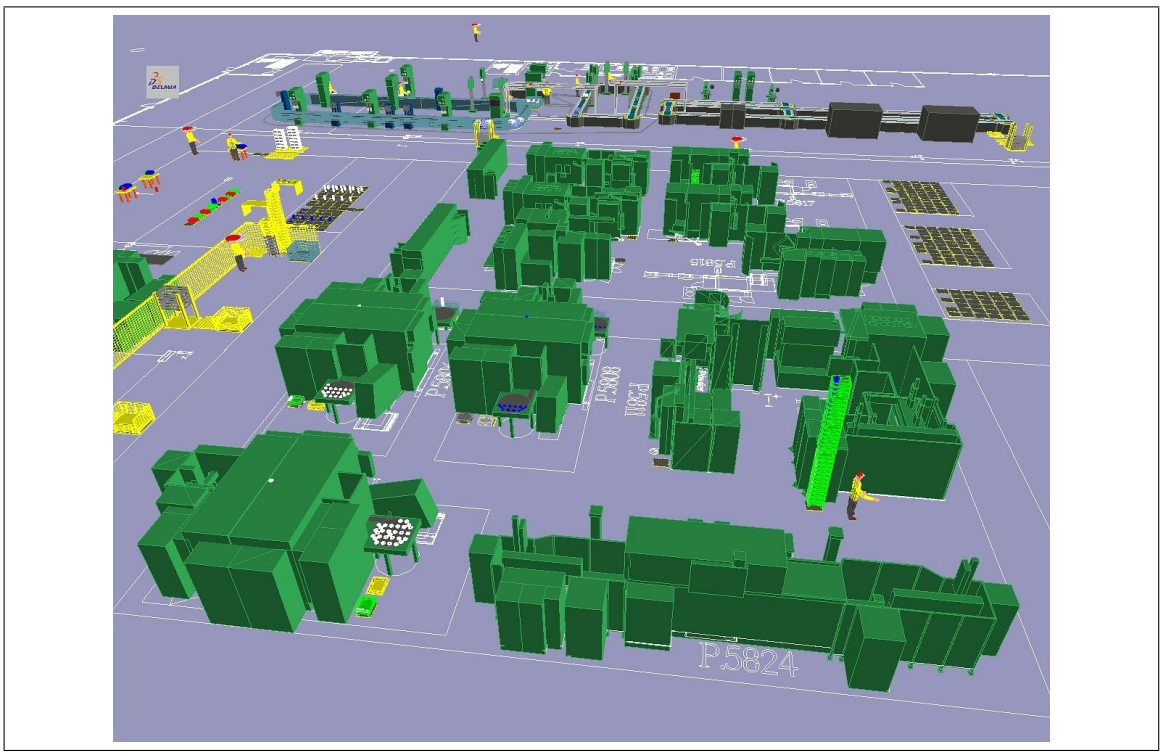


Abb. 2.6: Beispiel eines 3D-Modells in der Teilefertigung

Verteilung von Werkern auf Maschinengruppen, um nur ein paar Beispiele zu nennen. Der Maschinenpark in westlichen Produktionsländern ist aufgrund der hohen Fertigungslöhne stark automatisiert. Zur Be- und Entladung stehen Transportbänder, Greifersysteme und Roboter zur Verfügung, die einen Fertigungsprozess kontinuierlich mit Teilen versorgen. Die Maschinen selbst verfügen über Werkzeugrevolver, integrierte Messvorrichtungen und ausgeprägte NC-Steuerungen, welche bei hinreichender Teileverfügbarkeit eine unterbrechungsfreie, automatisierte Produktion erlauben. Diese fortgeschrittenen Technologien ermöglicht es, dass ein Werker im Rahmen seiner Tätigkeit und seiner Qualifizierung an mehreren Maschinen eingesetzt werden kann. In den Methoden der Zeitwirtschaft nach REFA [REF91] ist dieser Ablauf auch unter dem Begriff der Mehrmaschinenbedienung bekannt und wird in zeitwirtschaftlichen Berechnungen mit dem sogenannten Mehrmaschinenfaktor (MS) ausgedrückt.

Ein anderes, nicht zu unterschätzendes Merkmal der teilefertigenden Industrie ist die Teile- und Variantenvielfalt. Diese bedingen hohe Anforderungen an die Simulationssysteme selbst. So müssen beispielsweise in einem Produktionsbereich für den Sondergetriebebau im Werk Gaggenau regelmäßig ca. 7500 verschiedene Teilevarianten mit Jahresstückzahlen zwischen 1 und 150000 pro Jahr gefertigt werden. Dabei hat sich gezeigt, dass nicht jedes Simulationssystem wirtschaftlich mit Rechnerressourcen umgeht und somit nicht mit einer so großen Anzahl von Objekten im System akzeptable Leistung bringt. Auch bei der Modellierung selbst ist der Aufwand bei großer Variantenanzahl enorm, da alle technischen Zusammenhänge, wie z.B. die Baubarkeit bestimmter Variantenkombinationen, im Modell berücksichtigt und abgebildet werden müssen. Weiterhin müssen im Bereich der Teilefertigung eine große Anzahl verschiedener Da-

ten berücksichtigt werden, welche teilweise enorm wichtig für einen hohen Detaillierungsgrad und somit auch eine gute Realitätstreue sind.

Viele der simulationsrelevanten Daten sind in der Industrie bereits in diversen Produktions- und Planungssystemen vorhanden. Bei der Durchführung von Simulationsstudien fallen erfahrungsgemäß ca. 40% des zeitlichen Aufwands für die Beschaffung von Datenmaterial an [Wor02]. Im Rahmen dieser Arbeit soll der Diskrepanz zwischen dem Aufwand für die Datensammlung und der Tatsache, dass viele dieser Daten bereits in rechnertauglichem Format in den Systemen vorhanden sind, ein Stück entgegengewirkt werden.

2.4 Bedeutung digitaler Planungsmethoden im Planungsprozess

In den 90er-Jahren konnte das schnelle Wachstum und die rapide Steigerung der Durchdringung von CAD-Systemen in der Planung beobachtet werden. Heute sind Systeme, wie z.B. Pro Engineer oder Catia (beide CAD) in der Industrie nicht mehr wegzudenken. Die Welt ist dreidimensional und eben solches erwartet man auch von Systemen in der Produktentwicklung und der Produktionsplanung. Jedoch lässt sich beobachten, dass die Durchdringung von digitalen Planungsmethoden in der Produktentwicklung wesentlich höher ist als in der Produktionsplanung.

Während in der Entwicklung 3D-Systeme zu den absoluten Standardwerkzeugen gehören, wird in der Produktionsplanung häufig noch immer auf uneinheitlicher Basis (z.B. Papier, Excel, etc.) gearbeitet. Das führt zwangsläufig zu einer unzureichenden Ausschöpfung der heute vorhandenen, digitalen Hilfsmittel und somit auch zu einem großen, noch nicht ausgeschöpften Potenzial. Planungsfehler, mangelnde Kommunikation und uneinheitliche Datengrundlagen sind in größeren Industriebetrieben kein Fremdwort und die hierdurch entstehenden Mehrkosten und zeitlichen Mehraufwände sind immens.

Auch die Anforderungen in der Produktionsplanung haben sich in den letzten Jahren verändert. Während einst die Produktivitätsziele und verbunden damit die Gestaltung rationeller Fertigungs- und Montageverfahren vordergründig waren, konzentriert man sich heute mehr und mehr auf die Homogenisierung von Produktlebenszyklen und Produktionssystemen unter Einhaltung restriktiver Kostenziele [BB03].

Betrachtet man den Bereich der Produktionssteuerung, also den Teil der Produktentstehung ab dem „Start of Production“ (SOP), so kann man feststellen, dass auch in diesem die Durchdringung von Rechnersystemen sehr hoch ist. Man findet heute in der Produktion sehr ausgereifte Leitstandtechnik und Logistiksysteme. Produktionsabläufe nach dem MRP[2] II, Kanban- und ConWIP[3]-Prinzip werden von rechnergestützten Steuerungssystemen eingelastet, gesteuert und überwacht. Ein weiterer Bereich, in dem leistungsfähige IT-Systeme zum Einsatz kommen, ist die Steuerungstechnik von Maschinen (z.B. SPC), Montagesystemen (z.B. OPC) und die Offlineprogrammierung von Robotern. In den genannten Bereichen wurde in den vergangenen Jahren

[2] **M**anufacturing **R**esources **P**lanning
[3] **Con**stant **W**ork **I**n **P**rocess

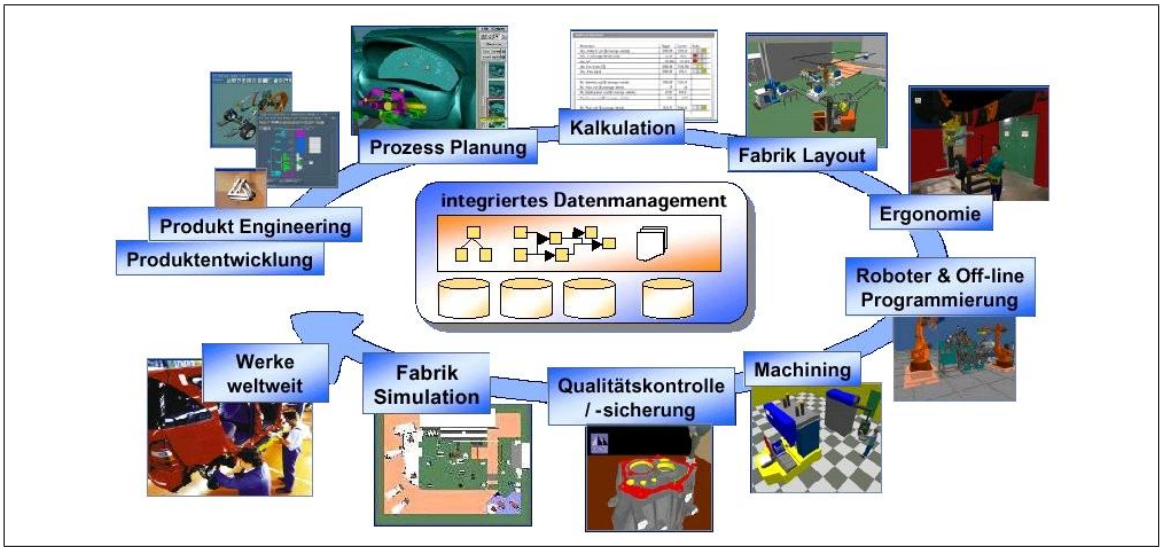


Abb. 2.7: Haupteinflussbereiche der digitalen Fabrik [SS02]

viel Geld für IT-Maßnahmen investiert, während bei fortschrittlichen Systemen für die Produktionsplanung nur zögerlich aufgerüstet wurde.

Von den digitalen Planungsmethoden erhofft man sich heute, dass mit ihrer Hilfe die Lücke in der leistungsfähigen Systemwelt im Bereich der Planung geschlossen werden kann. Es müssen viele unterschiedliche Stadien der Planung beachtet werden. Dazu zählen einerseits die Durchgängigkeit von der Grob- bis zur Serienplanung und andererseits die verschiedenen Gesichtspunkte, wie z.B. Materialfluss, Logistik, Ergonomie, etc.. Dabei ist für eine erfolgreiche Umsetzung eine durchgängige und einheitliche Datenbasis Grundvoraussetzung (vgl. Abb. 2.7). Sie bildet den Sockel für eine ganzheitliche, digitale Fabrikplanung und hilft gleichzeitig Kosten und Zeit durch Datenredundanz und -beschaffung zu vermeiden [RJ03].

In der Automobilindustrie gibt es unterschiedliche Ansätze die genannten Ziele zu erreichen. Insbesondere im Bereich der einheitlichen Datenhaltung sind sich jedoch alle einig. In Zeiten einer vernetzten Welt und leistungsfähiger Web-Technologien, wie man sie z.B. vom Onlinebanking, schnellen Suchmaschinen und weltweit verteilten Datenbanksystemen kennt, stehen effiziente technische Möglichkeiten zur Verfügung. Diese bieten einheitliche und ganzheitliche Datenhaltung und -zusammenführung auf kostengünstiger Basis und bilden somit einen weiteren Grundstein der Digitalen Fabrik.

Kapitel 3

Verteilte Datenbanken und Simulation

In der Anfangszeit der betrieblichen Datenverarbeitung waren noch alle Anwendungen dateibasiert, da es zu der Zeit noch keine Alternativen gab. Zur Speicherung von Daten verwendete man sequentielle Dateien (auch Kettendateien genannt), welche auf Magnetbändern und ganz am Anfang sogar auf Lochstreifen und Lochkarten gespeichert waren. Bezüglich der Flexibilität waren diese Ansätze jedoch unzureichend und kleine Änderungen bedurften sehr aufwändiger Erweiterungen und Ergänzungen der Dateistruktur. Um bestehende Anwendungen nicht zu gefährden, wurden meist nicht die Originaldateien modifiziert, sondern durch Umkopieren und Hinzumischen der neuen Dateien sogenannte „abgeleitete“ oder „abhängige“ Dateien gebildet. Im Laufe der Zeit bildeten sich komplexe Geflechte an Abhängigkeiten und die Konsistenzhaltung der Daten wurde immer schwieriger [Dad96].

In den 60er-Jahren kamen die ersten Datenbank-Management-Systeme (DBMS) auf den Markt, welche es ermöglichten, die Komplexität in der Anwendungsentwicklung zu reduzieren und somit auch die Fehlerquote zu verbessern. Die zunehmende Globalisierung der Märkte und die Vernetzung weltweit angesiedelter Produktionsstandorte führten zu der Notwendigkeit, Informationen jederzeit und überall verfügbar zu machen, unabhängig davon, wo die Daten anfallen oder verarbeitet werden [Kud92]. 1995 nahmen Lang und Lockemann an, dass die Zukunft der Informationslandschaft von großflächigen Netzen mit hunderttausenden von Knoten, an die sich Rechner unterschiedlichsten Leistungsvermögens, vom Großrechner über Hochleistungsstationen bis hin zu Desktops und Notebooks, die sich über alle Kontinente hinweg verästeln, geprägt sein wird [LL95]. Heute können wir sehen, dass diese Zukunftsvision innerhalb weniger Jahre zur Realität geworden ist, und dass die Möglichkeiten, welche die Zukunft noch bringen wird, heute noch gar nicht absehbar sind.

Mit den wachsenden Möglichkeiten, die unsere vernetzte Informationswelt mit sich bringt, wächst auch die Menge an gespeicherten Informationen, welche sowohl an einer Stelle konzentriert als auch über viele Knoten verteilt auftreten können. So großartig diese neuen Möglichkeiten auch sind, die Forderung nach einer effizienten Darstellung und Zusammenführung der Fülle an Daten wird gleichwohl wachsen. Schon heute kann man im Internet beobachten, dass es häufig gar nicht so einfach ist, die Informationen zu finden, die man eigentlich sucht!

Innerhalb einer rechnerintegrierten Fertigung, welche in der Industrie zum Standard geworden ist, spielt die Bereitstellung und Organisation von Daten eine zentrale Rolle bei der Umsetzung kurzer Durchlaufzeiten und hoher Qualitätsansprüche. Organisatorische, planerische, personelle und technische Maßnahmen müssen in wohlabgestimmter Weise zusammenwirken. Dabei

muss sichergestellt werden, dass alle Einzelschritte von der Produktentstehung, der Konstruktion über die Kalkulation bis hin zur Fertigungsvorbereitung, Materialflussplanung und Fertigungssteuerung reibungslos ineinander greifen. Sie müssen mit den nötigen Informationen versorgt werden, selbst wenn die einzelnen Schritte über lokal voneinander getrennte Standorte hinweg verteilt sind. Organisatorische Maßnahmen haben dafür Sorge zu tragen, dass ein Auftrag alle entsprechenden Funktionsketten ohne unnötige Verzögerungen durchläuft. Planerische Maßnahmen müssen die rechtzeitige Bereitstellung der Betriebsmittel sicherstellen. Personelle Maßnahmen sorgen für ausreichendes, entsprechend geschultes Personal und den technischen Maßnahmen obliegt es, für jeden Schritt sicherzustellen, dass dieser volle Informationen über die ihn betreffenden Entscheidungen in früheren Schritten und ebenso ausreichend Informationen über die Anforderungen nachfolgender Schritte erhält [LL95].

Die Umsetzung dieser Anforderungen wird häufig unter dem Dachbegriff *Integration betrieblicher Informationssysteme* zusammengefasst [Bar03]. Die integrierte Fertigung ist also auf das Vorhandensein einer leistungsfähigen Infrastruktur angewiesen, die für Informationsaustausch und -verarbeitung sorgt. Sie wird heute durch Rechnernetze, bestehend aus Hierarchien von Rechnern unterschiedlichen Leistungsvermögens, die über sogenannte „Local Area Networks“ (LAN) und „Wide Area Networks“ (WAN) miteinander verbunden sind, erzielt. Dabei sind im Laufe der Zeit die Systeme mit der permanenten Verbesserung der Möglichkeiten gewachsen. Alte Systeme wurden auf schnellere Hardware portiert oder durch neuere abgelöst. Neue Systeme haben mehrere Altsysteme ersetzt oder Altsysteme wurden dahingehend modifiziert, dass sie mit neueren Systemen zusammenarbeiten können. So hat sich über die Jahre besonders an älteren Produktionsstandorten eine sehr inhomogene Infrastruktur an Steuerungs- und Informationssystemen gebildet. Zusammenhänge in der IT-Landschaft sind häufig nur noch durch sehr komplexe Abhängigkeitsgraphen darstellbar und bei Änderungen an Hard- oder Software kommt es häufig zu schweren Integrationsproblemen mit existierenden Systemen.

Den Kern der meisten Steuerungs-, Informations- und Kalkulationssysteme bilden Datenbanken, welche das Datenmaterial meist in relationaler, strukturierter Weise speichern. Daher kommt den Datenbanken in der heutigen, vernetzten Welt eine große Bedeutung zu. Im Vordergrund steht dabei eine effiziente Struktur, welche redundante Datenhaltung weitgehend verhindert, bei trotzdem minimalen Betriebskosten und optimaler Verfügbarkeit der Architektur. Im Folgenden werden die Schwerpunkte insbesondere auf Unternehmensdatenbanken gelegt. Dabei sollen die unterschiedlichen Begrifflichkeiten und die möglichen Ausprägungen näher beschrieben werden.

3.1 Definition und Begrifflichkeiten

Datenbanksysteme bilden in der Informatik ein wichtiges Teilgebiet der Informationssysteme. Man versteht darunter in groben Zügen den Teil eines rechnergestützten Informationssystems, der sich mit der Beschreibung der vorhandenen Daten, ihrer Verwaltung sowie dem Umgang mit und dem Zugriff auf diese befasst. Ein Informationssystem eines Unternehmens enthält dabei die zur Kontrolle und Steuerung dieses Unternehmens notwendigen Informationen, respektive die zugehörigen Vararbeitungsprozesse [SS83].

3.1.1 Datenbanken und Datenbank-Management-Systeme

Das Konzept einer Unternehmensdatenbank, in der Fachliteratur auch häufig *enterprise database* genannt, kann sehr komplex sein. Die Definition dieses Begriffs kann nach Hackathorn in eine weitläufigere und eine engere Beschreibung untergliedert werden. Im verallgemeinerten Sinn ist eine Unternehmensdatenbank [Hac93]:

> *„the collection of any data that can affect decisions or can be affected by decisions within the enterprise. In effect, the enterprise database is the reflexion (or model) of reality that the enterprise perceives. This data can be in many forms. It can be formal (e.g. residing in a relational database on the mainframe) or informal (e.g. prior experiences of a manager). It can be structured (e.g. in a tabular form) or unstructured (e.g. in letters and memorandums). To leverage the investment in information technology, we must convert the relevant data to a formal and structured form.“*

In Bezug auf die Datenverarbeitung und auf Informationssysteme, wie sie später für die Simulation genutzt werden sollen, kann diese Definition jedoch weiter eingeschränkt werden. Folglich ist eine Unternehmensdatenbank [Hac93]:

> *„the formal data that resides in a database management system on some platform within the information system. The database could be designed using either the relational or object-oriented data models, as long as there is some level of database connectivity within the enterprise system. In other words, the database must be able to be shared across the enterprise.“*

Eine Datenbank ist, um es in kurzen Worten zusammenzufassen, ein mehr oder minder strukturierter Speicher für Informationen auf einem Rechnersystem. Um neue Daten in eine Datenbank einzupflegen, vorhandene Daten zu ändern oder zu löschen und insbesondere auch zum Suchen von Informationen innerhalb einer Datenbank bedient man sich eines DBMS. Man versteht darunter eine Software zur Organisation und Nutzung großer Datenansammlungen. Das erste allgemeine DBMS wurde Anfang der 60er-Jahre von Charles Bachman bei General Electric entwickelt und unter dem Namen *Integrated Data Store* bekannt. Es bildete die Grundlage für das Netzwerk-Datenmodell, welches von der Conference on Data Systems Languages (CODASYL) standardisiert wurde und fortan die Weiterentwicklung von Datenbanksystemen prägte [RG00]. Der Einsatz von DBMSen hat wesentliche Vorteile gegenüber anderen möglichen Ansätzen, wie einem Dateisystem (vgl. [RG00, SS83]):

- Datenunabhängigkeit
- effizienter Datenzugriff,
- Datenintegrität und -sicherheit
- Datenadministration
- Mehrfachzugriff und Datenwiederherstellung nach Fehlern
- reduzierte Anwendungsentwicklungszeit

Bisher beziehen sich die Definitionen und Begrifflichkeiten lediglich auf abgeschlossene, autonome Datenbanksysteme. Zur automatisierten Zusammenführung der simulationsrelevanten Daten aus betrieblichen Informationssystemen oder Datenbanksystemen, muss man tiefer in den Bereich des Zusammenspiels verschiedener, unabhängiger Datenbanken einsteigen. Wie bereits erwähnt, bestehen in Industriebetrieben komplexe Zusammenhänge zwischen verschiedenen Systemen und damit insbesondere auch ein permanenter Datenaustausch zwischen unterschiedlichen Datenbanken.

3.1.2 Konzepte und Strukturen von Datenbanksystemen

Es ist deutlich geworden, welche Vielzahl und Vielfalt bestehender und zukünftiger Möglichkeiten sich für Anwender und Anwendungsprogramme durch die Datenbanktechnologien erschließen. Gemeinsame Daten, welche einen bestimmten Umfang an Wissen repräsentieren, stellen das Bindeglied, die Datenbanktechnik das für das Zusammenwirken erforderliche Instrumentarium dar. Bedienen sich mehrere Anwender eines gleichen Datenbestandes, so spricht man von *Kooperation*. Änderungen die von einem Nutzer gemacht werden, sind gleichzeitig für alle anderen Nutzer auf derselben Datenebene sichtbar. Ist der Teil der Daten, die gemeinsam und auch gleichzeitig genutzt werden, in großem Maße überlappend, so spricht man auch von *Integration* der Datenbestände. Diese findet man auch in der rechnerintegrierten Fertigung. Den weniger häufig auftretenden Fall, dass Datenbestände größtenteils isoliert betrieben werden und die Überlappung der gemeinsam genutzten Daten zu jedem Zeitpunkt gering bleibt, nennt man auch *Koordination* [LL95].

Die Kooperation von Datenbanksystemen ist maßgeblich durch die Entwicklung im Bereich von Prozessoren, Hintergrundspeicher und insbesondere der Datenübertragungssysteme geprägt. Während in der Vergangenheit noch Systeme isoliert betrieben wurden und der Austausch unter verschiedenen Datenspeichern noch eher nebensächlich war, so sind heute zusammengeschaltete Dienste, wie man sie von Flugbuchungs- und Aktienhandelssystemen kennt, nicht mehr aus dem täglichen Geschäftsleben wegzudenken. Eine weitere Veränderung über die Zeit hat sich im Bereich der angebotenen Dienste und Funktionalitätsprofile von Datenbanksystemen ergeben. In der Vergangenheit beschränkte man sich noch auf einige wenige, vordefinierte Dienstangebote, während sich die Datenbanktechnologie heute bemühen muss, jeder Anwendung ein funktionalitätsgerechtes Profil anzubieten. Das Dienstangebot und das Funktionalitätsprofil werden häufig mit dem Begriff des Datenmodells beschrieben [LL95]. Folgen die Dienste der beteiligten Datenbanksysteme ein und demselben Datenmodell, so definiert man das System als *homogen*. Sowohl die Clients als auch der Server verwenden dabei das gleiche DBMS. Datenbanksysteme, bei denen unterschiedliche Datenmodelle zusammenwirken nennt man *heterogen*. Das bedeutet, dass Datenbanksysteme verschiedener Hersteller über Gateways miteinander verbunden werden (vgl. Abb. 3.1) [Rol03]. Die Ausprägung der Heterogenität ist sehr vielschichtig und reicht von unterschiedlichen Betriebssystemen, bis hin zu Abweichungen in den Wertebereichen bestimmter Attributwerte (vgl. Abb. 3.2).

Heterogen verknüpfte Datenbanken werden in beinahe allen Bereichen des Handels, der Industrie und des täglichen Lebens eingesetzt. Ein Beispiel dafür ist die Suche eines Buches über mehrere Bibliothekskataloge hinweg innerhalb einer einzigen Anfrage. Die Simulation, als „Kunde“

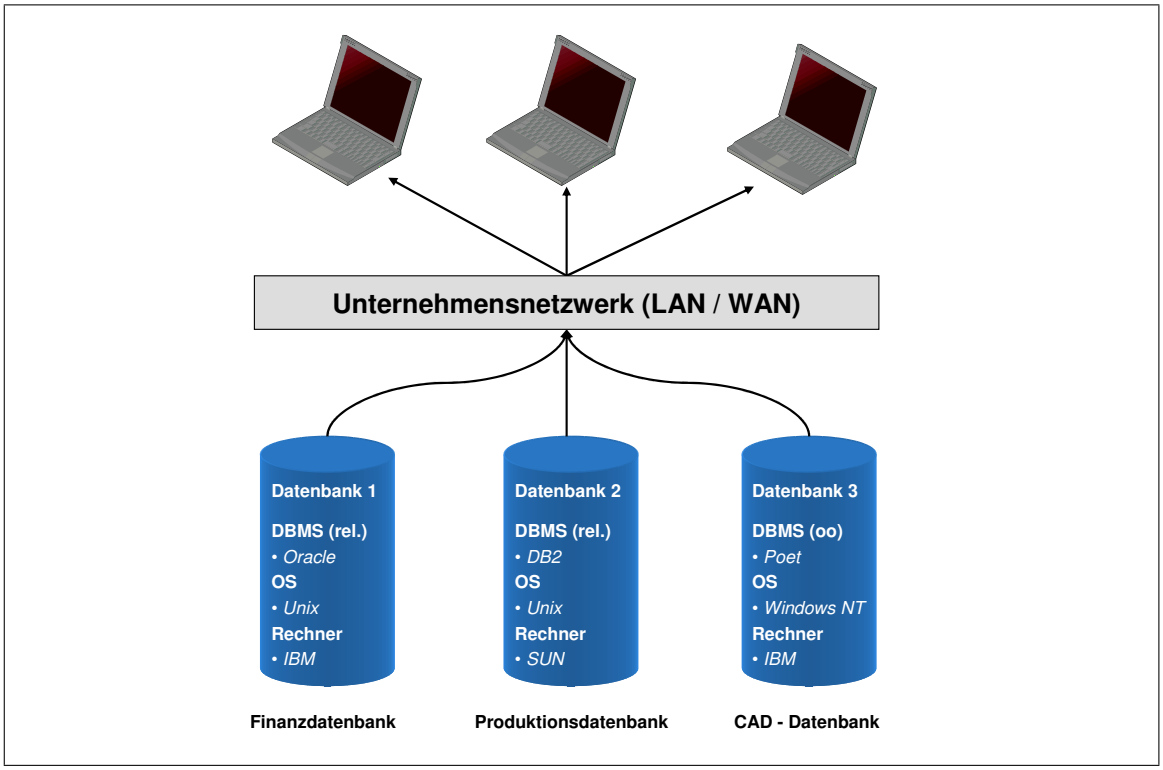


Abb. 3.1: Beispiel für heterogene Datenbanken

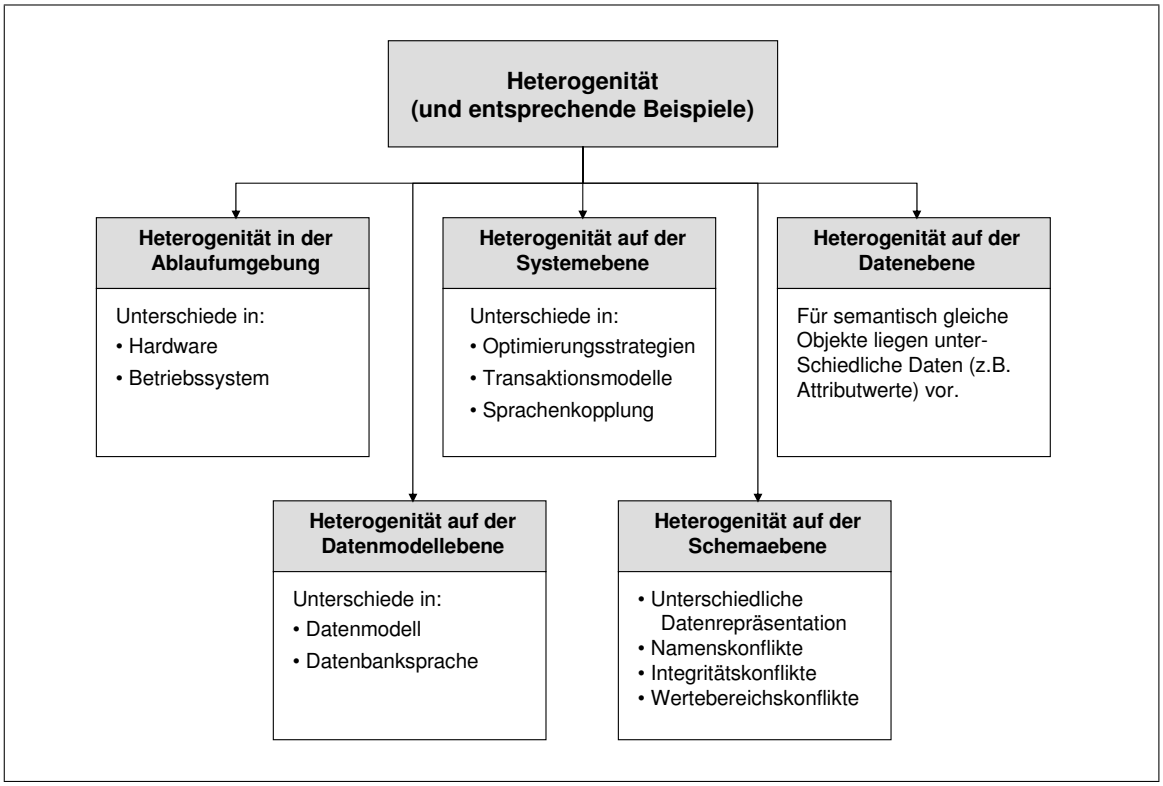


Abb. 3.2: Ausprägungen von Heterogenität bei Datenbanksystemen (nach [Rah94])

von Informationen aus unterschiedlichsten Systemen, muss folglich in eine systematische Verknüpfung und Zusammenführung von Daten aus heterogenen Datenbanksystemen eingebettet werden. Diese Bündelung heterogener Informationsquellen erzeugt jedoch häufig Schwierigkeiten, welche es zu beachten und zu vermeiden gilt. Während bei homogenen Systemen die Zusammenstellung von Daten aufgrund des abgeschlossenen Datenmodells und den einheitlichen Kriterien der Datenhaltung Missverständnisse leichter auszuschließen sind, muss bei kooperierenden, heterogenen Systemen die Datenzusammenführung im Nachhinein entwickelt werden. Es gilt die Spezifika der unterschiedlichen Datenmodelle zu beachten und Fehlermöglichkeiten, wie sie durch über Jahre hinweg entstandene „Altlasten“ in den Datenbanken entstehen können, weitestgehend auszuschließen.

Im Bereich der Kooperationen bei den verteilten Datenbanksystemen kann man zwei sehr unterschiedliche Funktionalitäten unterscheiden. Es ist denkbar, dass man aufgrund der Datenmenge ein einzelnes System auf mehrere verschiedene verteilt, um eine Lastverteilung unter den Datenbanken zu erreichen und somit steigenden Anwendungsanforderungen gerecht zu werden. Im Folgenden soll jedoch von dem Szenario ausgegangen werden, dass in den simulationsrelevanten Datenbanksystemen kein globales Schema und Datenmodell vorhanden ist und folglich die diversen Systeme noch nicht auf einen globalen Datenaustausch abgestimmt sind. Diese verteilten Datenbanken sind typische Vertreter einer auf Integration bedachten Kooperation.

3.1.3 Arten verteilter Datenbanksysteme

Ein verteiltes Datenbanksystem (VDBS) ist eine Sammlung mehrerer, logisch zusammenhängender Datenbanken, verteilt innerhalb eines Rechnernetzwerks. Ein verteiltes Datenbankmanagementsystem (VDBMS) ist somit ein Softwaresystem, welches das Management eines verteilten Datenbanksystems regelt und die Verteilung gegenüber dem Anwender transparent macht [ÖV99]. Um ein VDBS zu bilden genügt es nicht, dass Dateien logisch zusammengehören, sondern es muss auch eine Struktur innerhalb der Dateien vorhanden sein, welche den Zugriff über eine allgemeingültige Schnittstelle erlaubt. Die physikalische Verteilung von Daten über mehrere Orte ist im Bereich der verteilten Systeme sehr wichtig, führt aber auch zu Schwierigkeiten, welche man bei lokalen Datenbanken nicht vorfindet. Die geografische Verteilung spielt dabei nur eine untergeordnete Rolle, denn man spricht auch von einem VDBS wenn zwei Datenbankrechner innerhalb ein und desselben Raums stehen. Andererseits genügt es der Definition eines VDBS nicht, wenn nur eine einzige Datenbank auf einem Knoten innerhalb eines Netzwerks vorhanden ist und von verschiedenen Orten aus genutzt wird. In diesem Fall unterscheidet sich die Problemstellung des Datenbankmanagements nicht von der eines zentralen, alleinstehenden Datenbanksystems. Der einzige Unterschied, welcher in dieser Variante auftritt, sind Übertragunsverzögerungen. Es ist logisch, dass nur das Vorhandensein eines Rechnernetzwerks, oder einer Ansammlung von Dateien nicht ausreicht, um ein verteiltes Datenbanksystem zu formen [Abe03].

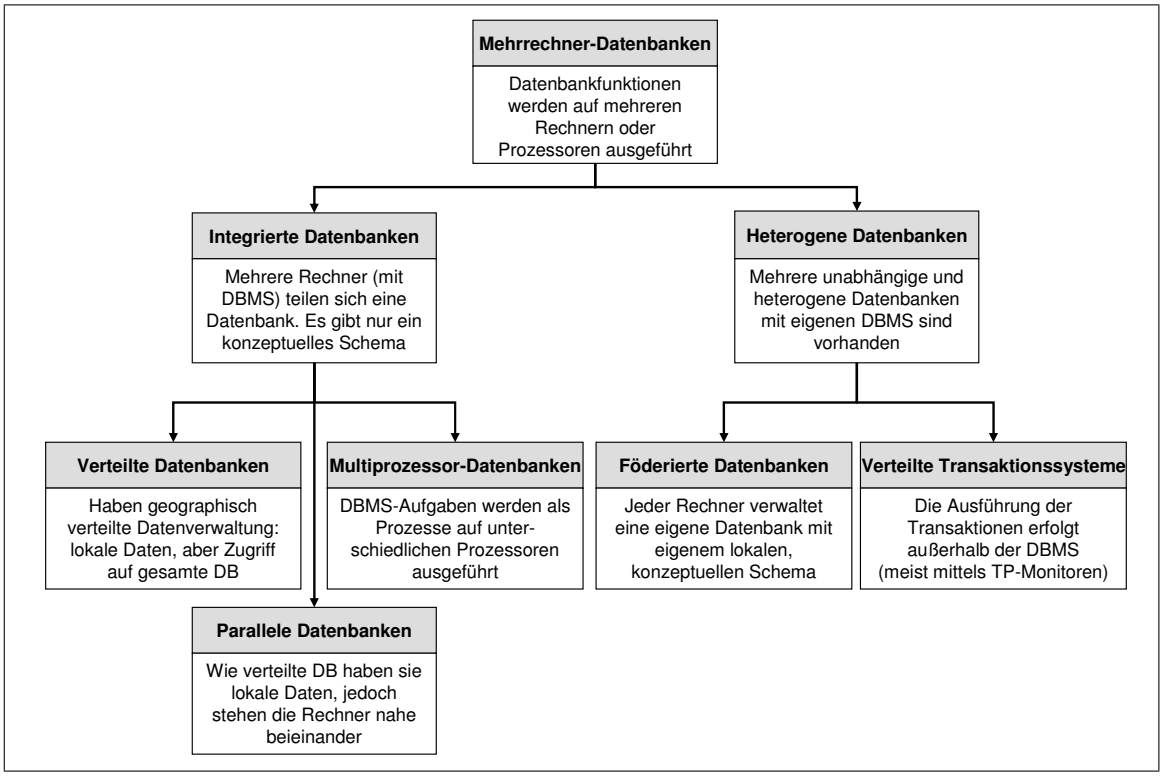


Abb. 3.3: Klassifikation von Mehrrechner-Datenbanksystemen (nach [Rah94])

3.1.4 Kopplungsmöglichkeiten von Multidatenbanksystemen

Die Möglichkeiten zur Kopplung von lokal verteilten und in sich autonom agierenden Datenbanksystemen können ganz erheblich variieren. Es muss entschieden werden, wie Prioritäten in der Auftragsbearbeitung festgelegt werden. So gibt es die Möglichkeit globale Anwendungen vor lokalen und umgekehrt zu favorisieren, oder aber zumindest beide im Vorrang gleichzusetzen. Von dieser Festlegung hängt unter anderem der Eingriff in die Autonomie der lokalen Systeme ab und es kann zu Verlusten an Effizienz führen. Die unterschiedlichen Kopplungsgrade führen zu einer Reihe unterschiedlicher Koordinierungsformen [ÖV99].

3.1.4.1 Kopplung mittels globaler Schemata

Bei der Kopplung von Multidatenbanksystemen mittels globaler Schemata unterscheidet man zwischen [LL95]:

- Verteilten Datenbanken
- singulären Föderationen
- multiplen Föderationen

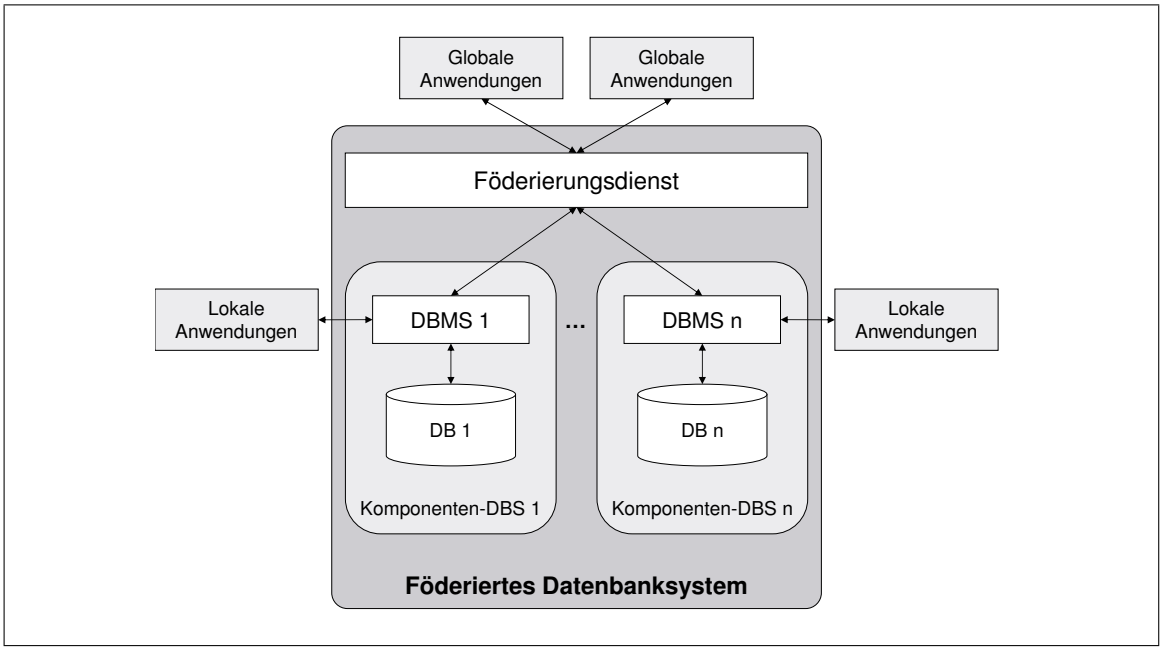


Abb. 3.4: Allgemeine Architektur föderierter Datenbanksysteme (nach [Con97])

Hauptmerkmal verteilter Datenbanken ist eine vollständige Ortstransparenz der unterliegenden Speicherorte. Der Datenabnehmer stellt seine Anfrage an ein zentrales Managementsystem und überlässt die Beschaffung der Daten an den unterschiedlichen Orten diesem System. Es existiert aus Nutzersicht quasi nur ein einziges Datenbanksystem und eine Datenbasis oder auch ein Schema. Die unterliegenden lokalen Datenknoten besitzen folglich nur eine sehr geringe Autonomie.

Eine Lockerung dieser Bedingungen findet man bei singulären Föderationen. Die lokalen Datenbanksysteme dürfen heterogen sein und somit auf unterschiedlichen Datenmodellen basieren. Allerdings unterwerfen sich diese den Vorgaben eines globalen Schemas und müssen sich folglich mit anderen Systemen im Netz abstimmen. Lokale Anfragen gehen von lokalen Schemata aus, globale Anfragen beziehen sich hingegen auf das globale Schema (vgl. Abb. 3.4). Die globalen Abfragen müssen wiederum auf lokale Datenbanksysteme umgesetzt und an diese weitergegeben werden. Es entsteht die Notwendigkeit von Abbildungsmechanismen zwischen globalem und lokalem Schema. Diese Art der Multidatenbanken erlaubt bereits die Integration weiterer lokaler Systeme ohne zusätzliche Eingriffe, da nur der Abbildungsmechanismus, also eine Art Übersetzer, hinzugefügt werden muss.

Die Bedingungen bei singulären Föderationen werden bei der Definition von multiplen Föderationen noch weiter gelockert. Man unterstellt bei diesen, dass nicht jeder Knoten in einem Netz mit jedem anderen zusammenwirken muss, und dass viele Daten von rein lokaler Bedeutung und Zugänglichkeit sind. Es kann daher auch kein globales Schema mehr existieren. Zwei Knoten, welche Daten untereinander austauschen wollen, müssen sich vorab über ein gemeinsames Schema absprechen. Dabei ist es üblich, dass jedes System innerhalb einer multiplen Föderation zwei Schemata unterhält. Ein Exportschema teilt jedem anderen Knoten mit, welche Daten von dem jeweiligen System global zur Verfügung gestellt werden. Das Importschema ist eines der

bereits erwähnten gemeinsamen Schemata und bezieht sich auf einen entfernten Netzknoten. Auch bei dieser Art der Kopplung ist eine Integration weiterer Systeme ohne Eingriffe möglich, es muss jedoch für jedes gemeinsame Schema ein Anfrageübersetzer hinzugefügt werden.

3.1.4.2 Rein sprachliche Kopplung

Sprachlich gekoppelte Multidatenbanken können, solange ein gemeinsames Schema existiert, durch Einrichtung von Übersetzern, auf den späteren Verbundbetrieb vorbereitet werden. Existieren keine gemeinsamen Schemata, so dürfen alle mit Anfragen in Verbindung stehenden Tätigkeiten auch erst zum Anfragezeitpunkt ausgeführt werden. Das bedeutet, dass die entsprechenden Anfragen auch alle zur Koordinierung notwendigen Angaben enthalten. Die Anfragen orientieren sich dabei an den Schnittstellen und somit auch an dem Schema der zugehörigen lokalen Datenbank. Es ist folglich Angelegenheit des anfragenden Datenbanksystems, seine Bedürfnisse in der Begrifflichkeit des angefragten Systems auszudrücken. Diese Art der Kopplung stellt die härtesten Ansprüche an das Kommunikationsvermögen lokaler Datenbanksysteme innerhalb einer Föderation und sie existiert daher in der genannten Reinform nicht [LL95].

3.1.5 Schemaklassen und Prozessoren bei föderierten Datenbanken

Im Folgenden soll zur Veranschaulichung der tieferen Zusammenhänge bei verteilten oder föderierten Datenbanksystemen eine vereinfachte Darstellung der Referenzarchitektur nach Sheth und Larson [SL90] bezüglich Schemaklassen und Prozessoren aufgezeigt werden. Die Referenzarchitektur kann zur einheitlichen Beschreibung verschiedener Systeme herangezogen und als methodisches Planungsinstrument für zukünftige Systeme verwendet werden. Die Architektur ist getrennt nach den auftretenden Schemata und den auf ihnen operierenden Prozessoren, welche für das Verständnis der Rolle der Schemata von Bedeutung sind.

3.1.5.1 Schemaklassen

Die Schemaklassen können grundsätzlich in vier unterschiedliche Arten unterteilt werden, wobei ein System durchaus mehrere Schemata gleichzeitig besitzen kann [Rah94, LL95] (vgl. Abb. 3.5):

1. Lokales Schema
2. Komponentenschema
3. Exportschema
4. Föderiertes Schema

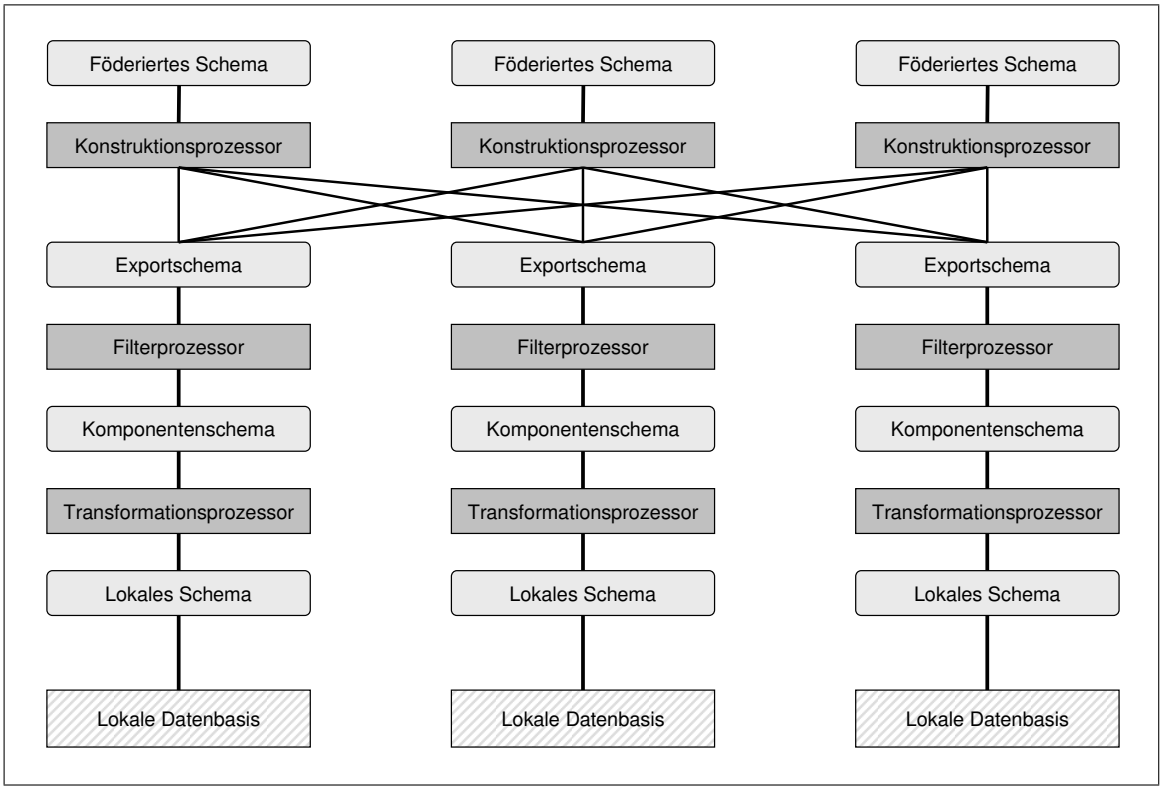


Abb. 3.5: Referenzarchitektur einer föderierten Datenbank [LL95]

Als *lokales Schema* bezeichnet man das Schema einer Datenbasis an einem einzelnen Netzknoten. Es wird ohne Berücksichtigung von anderen Netzknoten, also in voller Autonomie, erstellt und nutzt ausschließlich das Datenmodell des lokalen Datenbanksystems. Im Rahmen eines heterogenen Netzwerks, bestehend aus mehreren Datenbanken, wie es in der Industrie vorherrscht, können netzweit beträchtlich viele lokale Schemata auftreten.

Das *Komponentenschema* ist eine Erweiterung des lokalen Schemas um eine für die Koordinierung notwendige, respektive diese erleichternde Ergänzung. Es kann ein netzweit vereinbartes Datenmodell ergänzt sein, welches nach den Regeln der Abbildung zwischen Datenmodellen durch Übersetzung des lokalen Schemas erzielt werden kann, sofern das lokale Schema vom Komponentenschema abweicht. Dabei kann das Komponentenschema zusätzliche Angaben beinhalten, welche im lokalen Schema nicht vorkommen, eine Koordination jedoch erleichtern.

Sollen nicht alle Daten einer lokalen Datenbasis innerhalb einer Föderation zur Verfügung gestellt werden, so muss ein *Exportschema* vorgesehen werden, welches nur einen Ausschnitt aus dem Komponentenschema nach außen sichtbar macht. Außerdem kann dieses Schema zusätzliche Angaben über Rollen und Zugriffsrechte der entsprechenden Netzteilnehmer enthalten.

Die Integration aller Exportschemata der lokalen Datenbanken innerhalb einer singulären Föderation nennt man ein *föderiertes Schema*. Dieser Begriff besitzt auch bei multiplen Föderationen Gültigkeit, jedoch werden hierbei nur die Exportschemata der zusammenwirkenden Systeme innerhalb eines Netzes integriert.

Für die Simulation benötigt man aus mehreren Datenbanksystemen heterogenen Typs jeweils einen kleinen Ausschnitt der insgesamt verfügbaren Datenumfänge. Diese kommen aus Systemen, welche zumindest ein lokales Schema, in einigen Fällen auch ein Komponenten- und Exportschema besitzen. Jedoch sind diese für andere Anwendungsfälle erstellt worden und damit nicht für die Zusammenführung simulationsrelevanter Daten geeignet. Daher müssen im weiteren Verlauf der Arbeit ein Datenmodell erstellt und die zugehörigen Exportschemata definiert werden.

3.1.5.2 Prozessoren

Auf den verschiedenen Schemata operieren diverse Prozessoren mit unterschiedlichen Aufgabenverteilungen. Die Prozessoren werden in drei unterschiedliche Kategorien eingeteilt [SL90]:

- Transformationsprozessoren
- Filterprozessoren
- Konstruktionsprozessoren

Auf der untersten Ebene befindet sich der *Transformationsprozessor*, welcher die Umsetzung von Anfragen und Daten von einer Quellsprache, bzw. einem Quellformat in eine Zielsprache oder ein Zielformat übernimmt. Diese Aufgabe fällt vornehmlich beim Wechsel zwischen Schemata an, also insbesondere auch beim Übergang von lokalem Schema zu Komponentenschema. Dabei kann es bei der Transformation zu einem Wechsel des Datenmodells kommen.

Filterprozessoren werden anderen Prozessoren vorangestellt um sicherzustellen, dass nur zulässige Anfragen bearbeitet werden. Ihre Grundlage bilden sogenannte Zwangsbedingungen, die auf Anfragen und Daten definiert sind. Typische Beispiele sind Syntax- und Konsistenzprüfung.

Wichtig für die Verteilung sind die *Konstruktionsprozessoren*. Sie entscheiden bei Anfragen aus einem Knoten an das Netz über die Knoten, an die die Anfrage weitergeleitet wird. Sie wiederholen gegebenenfalls die Anfrage und die mit ihr einhergehenden Daten. Weiterhin müssen Konstruktionsprozessoren in umgekehrter Reihenfolge die Ergebnisse mehrerer Knoten zu einem einzigen Ergebnis zusammenstellen. Diese Prozessoren sind meist zwischen mehreren Exportschemata und dem föderierten Schema angesiedelt.

3.2 Dynamisch gewachsene IT-Systeme in Großunternehmen

Der Fortschritt bei den technischen Möglichkeiten von Hard- und Software haben in der Vergangenheit dafür gesorgt, dass sich in der Industrie stets neue und verbesserte Anwendungsbereiche für die rechnerintegrierte Fabrik ergeben haben. Dabei spielen die meist monolithischen DMBS eine zentrale Rolle. Mit dem Lauf des technologischen Fortschritts auf dem IT-Sektor kamen neue Systeme mit neuen Möglichkeiten zum Einsatz und wurden in die „Altwelt“ integriert.

Schnittstellen zwischen den Systemen wurden geschaffen und in täglichen, wöchentlichen oder monatlichen, meist nächtlichen Verwaltungsläufen werden Unternehmensdaten zwischen Datenbanken und anderen Anwendungen in allen nur erdenklichen Formen ausgetauscht. Dass hierbei Mängel in der Datenintegrität und -konsistenz auftreten, ist die logische Folgerung. Zusätzlich entsteht durch diese Methodik eine sowohl für die Effizienz als auch für die Betriebskosten nicht unerhebliche Datenredundanz, da gleiche Daten häufig in mehreren Systemen parallel gehalten werden.

Für den größten Teil der Produktions- und Informationssysteme wird die Bedeutung von Datenbanksystemen unverändert Bestand haben. Während die Industrie in den 80er-und 90er-Jahren noch vom Wirtschaftswachstum und einer zentralistischen Organisationsform geprägt war, kann man heute vermehrt kleinere, relativ autonome Einheiten erkennen. Es wird daher auch immer mehr Anwendungsbereiche geben, in denen sich der Trend von zentralen, monolithischen Datenbanksystemen weg zu übergreifenden, interagierenden Informationssystemen verschiebt. Informationssysteme heute und in der Zukunft bestehen aus einer Vielzahl in Netzwerken verteilter, technisch und semantisch heterogener Informationsquellen. Dies können Datenquellen (z.B. Daten- oder Wissensdatenbanken), Funktionen (z.B. Berechnungsbibliotheken oder Analysefunktionen) oder sogar komplette, bereits bestehende Anwendungssysteme sein [Kos99]. Dabei sind der Einsetzbarkeit fast keine Grenzen gesetzt. Sie erstreckt sich von der Integration mehrerer Komponenten komplexer Anwendungs- und Datenbanksysteme, wie Engineering-Umgebungen [Bül95, SBH93], über unternehmensweite Systeme, auch „Data Warehouses“ [Wid95] genannt, für Management-Informationssysteme, bis hin zu unternehmensübergreifenden Systemen. Die Integration und Interoperabilität solcher heterogener Informationsquellen wird eine zentrale Rolle im Rahmen der Einbettung digitaler Planungsmethoden in die heute vorhandenen IT-Strukturen spielen. Simulationsrelevante Daten müssen künftig aus den heterogenen, monolithischen Systemen zusammengeführt und in einer direkt für die Modellierung von Simulationsmodellen brauchbaren Form aufbereitet werden. Ein exemplarisches Framework zur Bewältigung dieser Aufgabe soll im weiteren Verlauf dieser Arbeit entwickelt und umgesetzt werden.

3.3 Möglichkeiten zur Integration heterogener Datenbanksysteme

Die Herausforderungen, welche durch die Einbindung heterogener Datenbanksysteme in die digitale Planungswelt entstehen, treten insbesondere bei Industriebetrieben auf, welche schon seit Jahren auf dem Markt sind. Bei diesen Unternehmen ist die IT-Landschaft in der Vergangenheit kontinuierlich gewachsen und hat sich dem technischen Fortschritt angepasst. Systeme verschiedener technischer Epochen führen eine Koexistenz und werden parallel zueinander eingesetzt. Nicht selten kommt es vor, dass sequenzielle Dateien und objektorientierte Datenbanksysteme gleichzeitig eingesetzt werden. Bei der effizienten Umsetzung von digitalen Planungsmethoden ist es jedoch unabdingbar, eine Integration der „Alten Welt“ in die „Neue Welt“ herzustellen. Es muss eine Vermittlungsebene zwischen den beiden Welten geschaffen werden, welche den bidirektionalen Austausch von Daten erlaubt, und welche gleichzeitig in der Lage ist, mit unterschiedlichsten Systemen zu kommunizieren und deren Informationen zu selektieren, respektive zusammenzuführen.

Die Möglichkeiten, welche von den einzelnen Systemen zur Integration zur Verfügung gestellt werden, sind so unterschiedlich, wie diese selbst. Geht man davon aus, dass es aus technischen und monetären Gründen nicht möglich oder wünschenswert ist, die Systeme selbst zu modifizieren, so muss man die Funktionalitäten, die standardmäßig zur Verfügung gestellt werden, nutzen. Aufgrund der Tatsache, dass die wenigsten Datenbanken in der Industrie autark arbeiten, kann man sich auch der Möglichkeit bedienen, notwendige Daten in Systemen zu holen, welche vom Primärspeicherort durch Datenaustausch versorgt werden. Dies kann insbesondere dann von Bedeutung sein, wenn eine spezielle Datenbank keine effiziente Integrationsmöglichkeit anbietet. Dabei muss jedoch stets die Problematik der Datenintegrität beachtet werden. Es spielt folglich eine große Rolle wann, wie oft und wie sich die einzelnen Systeme gegenseitig mit Daten versorgen.

In den vergangenen Jahren haben Unternehmen den Nutzen des World Wide Webs (WWW) für kooperatives Arbeiten und gesteigerte Effizienz beim Informationsaustausch entdeckt. Als Konsequenz werden immer mehr Systeme für die Nutzung im Intra- und Internet angepasst. Hierdurch entstehen neue Anforderungen bezüglich der Methoden und Hilfsmittel, welche den effektiven Zugriff auf Daten unterschiedlichster Formate und aus heterogenen Datenquellen ermöglichen [C$^+$03]. Das Ziel der Informationsintegration ist das Erschaffen einer globalen Beschreibung von Informationen, welche aus heterogenen Datenquellen stammen. Hierdurch soll der Nutzer mit einer einheitlichen Anfrageschnittstelle bezüglich der Quellen unterstützt werden, unabhängig von deren Herkunftsort und dem Grad der Heterogenität ihrer Daten. Es entsteht ein quasi globaler und einheitlicher Informationsraum, welcher für unterschiedlichste Anfragen und Zugriffe zur Verfügung steht [PS98].

In der Vergangenheit wurden verschiedene Ansätze und Hilfsmittel zum Umgang mit heterogenen Datenquellen entwickelt und es wurden Integrationsarchitekturen auf der Basis von Middleware-Datenmodellen vorgestellt [H$^+$99, LRO96]. In letzter Zeit entwickelt sich XML mehr und mehr zu einem Standard bei der Informationsverarbeitung und -bereitstellung im Netz. Als eine Konsequenz daraus wird bei webfähigen Informationssystemen immer mehr ein Szenario deutlich, bei dem XML zur Repräsentation von Austauschdokumenten, -daten und -strukturen adaptiert wird. In diesem Zusammenhang können leistungsfähige Integrationsmechanismen zur Organisation und zum Suchen von Informationen auf Basis von XML-Daten und XML-Strukturen entwickelt werden [D$^+$98, LRO96]. Die semantische Struktur in XML ermöglicht das effiziente Zusammenführen von Daten aus unterschiedlichen Quellen.

Im Folgenden sollen die Unterschiedlichen Möglichkeiten von Datenzugriffen auf Unternehmensdatenbanken aufgezeigt werden und deren Adaption in eine XML-Darstellung beschrieben werden.

3.3.1 Dateiexport und Deltadateien

Bei bestimmten Systemen in der Industrie, wie produktionskritischen Applikationen, bei deren Ausfall die Produktionssteuerung ausfällt, ist teilweise eine Echtzeitkopplung zu anderen Systemen aus Leistungsgründen nicht erwünscht. Diese Art von Informationsquellen stellen relevante Daten meist zu produktionsunkritischen Zeiten (Pausen, Wochenenden, nachts, etc.)

per Export zur Verfügung. Sie übertragen Daten mittels FTP[1] oder anderen Protokollen an sogenannte Fileserver. Diese dienen als Dateicontainer, welche Dateien in einer hierarchischen Struktur mehreren Clients zur Verfügung stellen. Dabei können entweder komplette Datenbestände auf einmal übertragen werden, oder es können Deltadateien exportiert werden, welche nur die veränderten Datensätze seit dem letzten Abzug beinhalten. Letzteres dient der Einsparung von Rechnerressourcen. So kann in vordefinierten Intervallen (z.B. einmal im Monat) eine Ausgabe des gesamten Datenbestands und dann einmal oder mehrmals täglich die Bereitstellung der veränderten Daten in einer Deltadatei erfolgen. Dabei kommen zur Strukturierung der Daten bekannte Formate, wie die Darstellung mit festen Spaltenbreiten, oder die semikolongetrennte Liste zum Einsatz.

Das Exportverfahren erschwert eine Echtzeitkopplung mit anderen Systemen erheblich, da das Lesen aus Dateien aufgrund der nicht vorhandenen Indizierung und Strukturierung sehr ineffizient ist. Bei der Anbindung an eine globale Datenstruktur muss also von Fall zu Fall unterschieden werden, ob die Exportdateien in ein sekundäres Datenbanksystem übertragen werden, oder ob aus der Datei direkt eine XML-Darstellung generiert wird, welche im Hauptspeicher eines Servers residiert. Letzteres kommt insbesondere dann in Frage, wenn es sich um kleine Datenbestände handelt, welche jedoch sehr häufig im Rahmen von Abfragen benötigt werden. Um einen effizienten Speicherzugriff zu ermöglichen, bieten sich Hashtabellen an, welche eine Indizierung der Daten nach einem oder mehreren bestimmten Schlüsseln erlauben. Ist weder die Datenübertragung in ein sekundäres Datenbanksystem, noch das Halten im Arbeitsspeicher erwünscht, so kann das strukturlose Textformat in eine XML-Struktur überführt und anschließend auf einem Datenträger gespeichert werden. Die XPath[2]-Systematik[Wik07i] ermöglicht ein gezieltes Springen durch XML-Dateien. Wenn die Daten in XML gut strukturiert gespeichert sind, kann somit effizient durch Rekursion in einem Suchbaum das gewünschte Element gefunden werden.

3.3.2 Batchverarbeitung

Eine aus der Zeit der Großrechnersysteme stammende und auch heute noch auftretende Methodik ist die Batchverarbeitung – auch Stapelverarbeitung genannt – von Anfragen. Dabei wird meist eine Anfrage im SQL-Format per FTP oder Dateisystem an einer bestimmten Stelle abgelegt, an der sich der Großrechner den Auftrag abholen kann. Dieser verarbeitet eine Anfrage nach der anderen und stellt die Ergebnisse als semikolongetrennte Liste wieder an einem für den Anfragenden zugänglichen Ort zur Verfügung.

Die Problematik bei der Online-Anbindung solcher Systeme liegt in der Zeitverzögerung zwischen Auftragserteilung und Ergebnisbereitstellung. Reaktionszeiten können im Bereich von Millisekunden liegen, bei gleichzeitigen, stärker priorisierten Verwaltungsläufen jedoch auch im Bereich von Minuten. Bei der Integration solcher Verfahren in eine verteilte Anfragestruktur muss beachtet werden, dass es Abhängigkeiten zwischen den Knoten gibt. Eine Anfrage über mehrere Server dauert mindestens so lange, wie der langsamste Teilknoten zum Antworten

[1] File Transfer Protocol
[2] XML Path Language

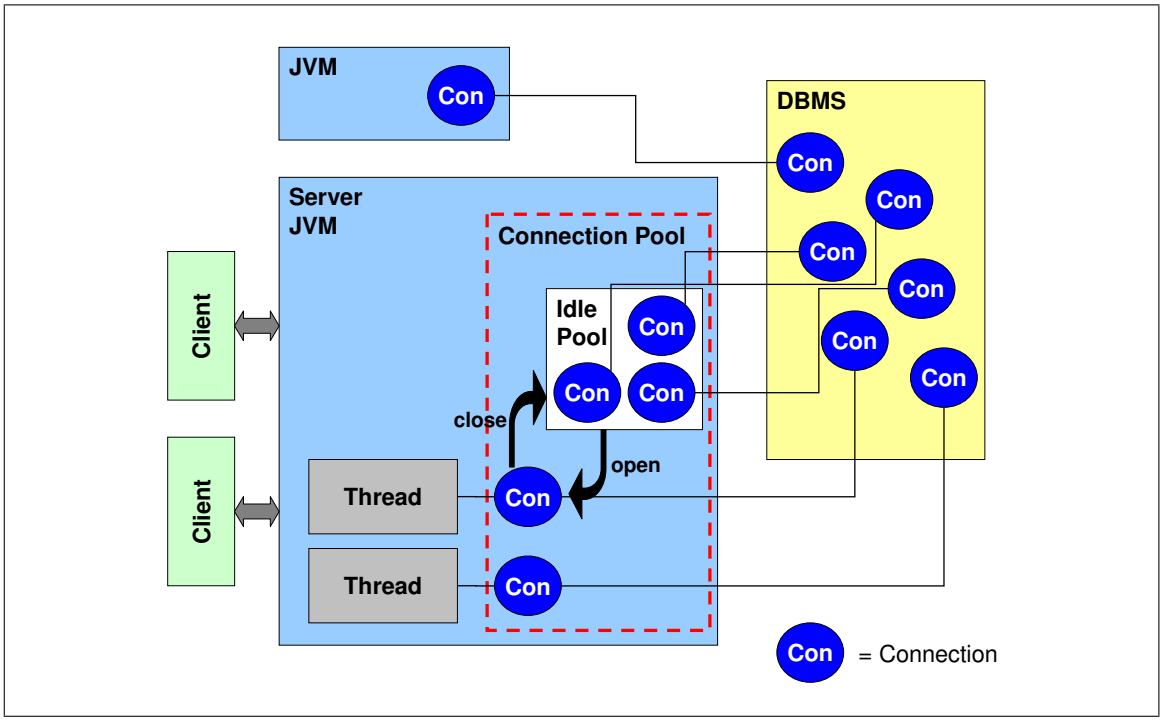


Abb. 3.6: Schematische Darstellung eines Connection-Pools (vgl. [Deh03])

braucht. Somit muss von Fall zu Fall abgewägt werden, ob es sinnvoller ist bestimmte Daten in einem Cache oder einem sekundären Datenbanksystem mit Online-Zugriff zwischenzuspeichern, oder ob die Wartezeit auf die Ergebnisbereitstellung aufgrund der Ergebnismenge in Kauf genommen wird. Handelt es sich z.B. um Jahresstückzahlen für alle Einzelteile einer Fabrik, welche nur einmal wöchentlich der Marktsituation angepasst werden, jedoch einen Datenumfang von mehr als 10.000 Sätzen ausmachen, so können die Daten durchaus in ein sekundäres Datenbanksystem eingespielt und wöchentlich automatisch angepasst werden. Soll jedoch ein Fertigungsleitstand in kurzen Zeitintervallen ein Simulationsmodell mit Daten versorgen, dann kann dieses Verfahren nicht angewendet werden. In diesem Fall muss nach Möglichkeit eine Echtzeit-Verbindung implementiert werden.

3.3.3 Echtzeit-Verbindung mittels Connection-Pooling

Die wohl effizienteste und komfortabelste Methodik zur Echtzeit-Zusammenführung von industriellen Datenbanksystemen stellt das sogenannte Connection-Pooling dar. Ein Datenbanktreiber stellt eine gewisse Anzahl von Verbindungen zu einem DBMS her und regelt die Kommunikation (vgl. Abb. 3.6).

Benötigt ein bestimmter Web-Dienst Daten aus einem Datenbanksystem, so wird ihm eine Verbindung aus dem Pool zugewiesen. Die Daten werden über diese Verbindung angezogen und nach Fertigstellung wird die Verbindung wieder in das Pool zurückgegeben. Sie kann fortan wieder von anderen Diensten genutzt werden (vgl. Abb. 3.6). Der große Vorteil bei dieser Technik liegt in dem wesentlich geringeren Kommunikationsaufwand zwischen Treiber und DBMS,

da nicht für jede Anfrage ein Benutzer und Kennwort ausgetauscht und geprüft werden muss. Außerdem können verschiedene Dienste bis zur eingestellten maximalen Anzahl von Verbindungen parallel auf eine Datenbank zugreifen. Das Connection Pooling wird heute in fast allen Diensten genutzt, welche einen regen Datenaustausch mit einem Datenbanksystem betreiben.

3.4 Autonome Datenbanksysteme und Simulationsanwendungen

Ein Sonderfall bei der Kopplung von heterogenen Datenbanksystemen und Simulationsanwendungen ist der nur lesende Zugriff auf Daten. Das hat zur Folge, dass man bei der Vernetzung mehrerer Datenbanken keine Transaktionskontrolle berücksichtigen muss, welche beim Schreibzugriff die Konsistenz der Daten sicherstellt. Außerdem kann dadurch eine wesentlich schnellere Kopplung erreicht werden, da der Kommunikationsaufwand, der für gewöhnlich bei Transaktionskontrollen anfällt, nicht notwendig ist.

3.4.1 Kopplung auf Basis mediatorbasierter Informationssysteme

Eine Möglichkeit zur Zusammenführung mehrerer heterogener Systeme auf Basis der in Abschnitt 3.3 dargestellten Methoden ist die Systematik der mediatorbasierten Informationssysteme. Diese zeichnen sich durch zwei spezifische Merkmale aus:

- Mediatorbasierte Informationssysteme stellen eine spezielle Form einer *eng gekoppelten Föderation* dar.
  $\Rightarrow$ es existiert ein global verwendbares Schema (Mediatorschema).
- Die Föderation erlaubt *nur lesende Zugriffe*.
  $\Rightarrow$ das Mediatorschema muss die lokalen Schemata nicht vollständig enthalten.

Ein Mediator stellt eine „Zwischenschicht“ zwischen Clients und Servern dar, die eine Kommunikation zwischen unterschiedlichen Knoten zulässt (siehe Abb. 3.7). Zu den Aufgaben des Mediators gehören:

- Anfragebearbeitung,
- fokussierender Zugriff auf verschiedene Quellen,
- Zusammenfügen von Informationen (Fusion),
- Aufbereitung von Informationen,
- Anpassung an technische Bedingungen der Nutzer.

Mediatoren erhalten Informationen und stellen diese für andere Mediatoren, Hüllklassen (Wrapper) oder externe Benutzer bereit. Sie stellen damit Sichten der in einer oder mehreren Quellen gefundenen Daten dar. Es gilt hierbei zu beachten, dass einem Mediator Anfragen gestellt werden, obwohl er keine Daten beinhaltet. Dieser muss darauf folgend die Informationen zur Beantwortung der Anfrage aus den Quellen zusammensuchen und dem Nutzer bereitstellen [Ull97].

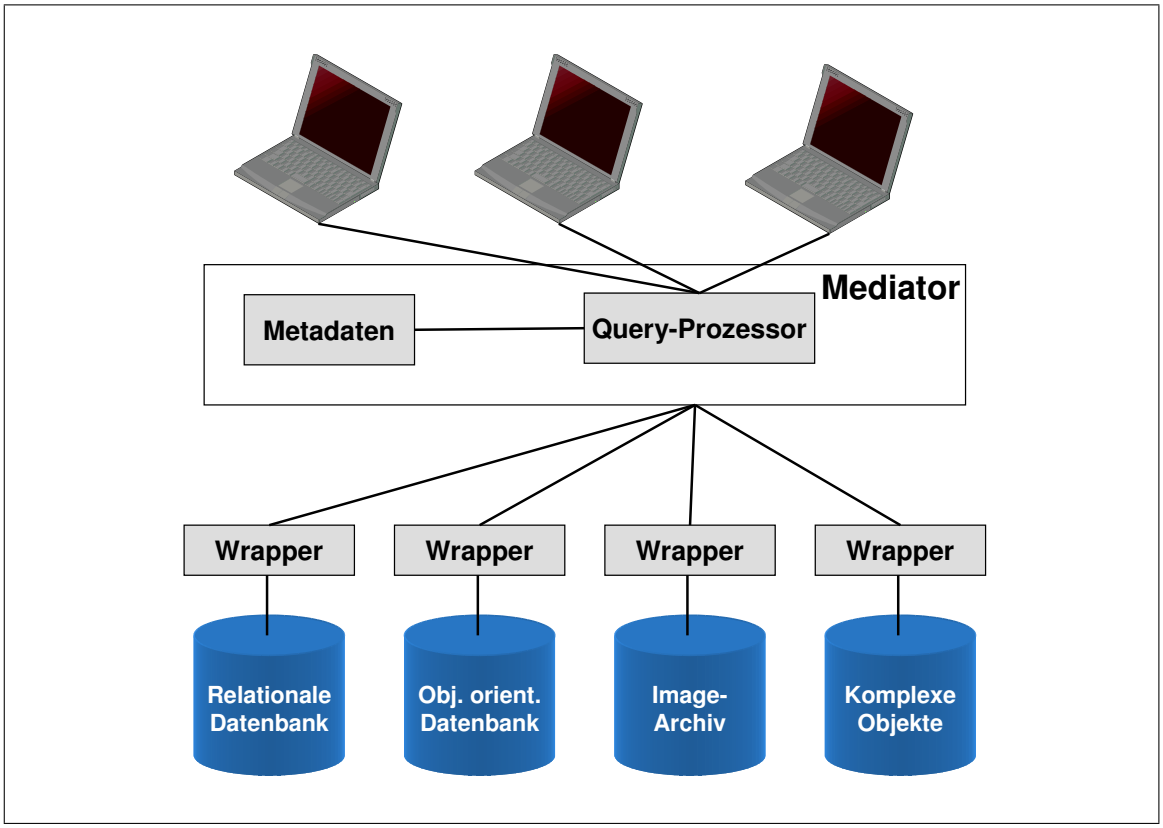


Abb. 3.7: Beispielarchitektur für ein mediatorbasiertes Informationssystem (nach [Ull97])

Die Clients stellen ihre Anfrage an den sogenannten Query-Prozessor innerhalb des Mediators, welcher das globale Schema und die Metadaten kennt. Die Metadaten enthalten die Beschreibung der Attributausprägungen, Wertebereiche und Relationen in den einzelnen Datenbanksystemen. Der Query-Prozessor kennt somit die Speicherorte und Zusammenhänge der gewünschten Daten und verteilt die Anfrage der Clients auf die entsprechenden Systeme, bzw. deren Wrapper (vgl. Abb. 3.7). Bei der Fusion der Informationen werden Redundanzen und Fehler anhand von Regeln gefiltert. Anschließend werden die gefundenen Informationen in eine für den Client verständliche Form konvertiert und den technischen Bedingungen und Möglichkeiten des Clients angepasst.

Die Wrapper-Technologie dient der Integration von Datenquellen unterschiedlichster Art. Dabei bilden diese eine Schnittstelle zu einer einzelnen Datenquelle mit ihrem entsprechenden Schema, ihrer Anfragesprache und ihrem Interface. Aufgrund der heterogenen Beschaffenheit der Datenbanksysteme wird für jedes System ein entsprechender Wrapper zur Verfügung gestellt. Dieser stellt die bidirektionalen Umsetzung von Quellformaten in Zielformate sicher und nimmt somit die Funktion des Transformationsprozessors aus Abschnitt 3.1.5.2 wahr.

3.4.2 Ausführungsoptimierung mittels Thread-Pools

Eine weitere Möglichkeit zur Verbesserung der Leistung bei Anfragen über verteilte Datenbanksysteme stellt ein parallelisierter Anfrageprozess dar. In Abschnitt 3.3.3 sind bereits die Vorteile eines Verbindungspools zu einer autonomen Datenbank beschrieben. Die Anfrageoptimierung mit dem Bucket-Algorithmus führt zusätzlich zu einer Minimierung der Anzahl an Anfragen über mehrere Systeme hinweg. Je nach Datenumfängen, die über ein mediatorbasiertes Informationssystem angefragt werden, und der entsprechenden Häufigkeit von Anfragen, kann es vorkommen, dass gleiche mehrfach ausgeführt werden. Diese sind zwar vom Inhalt nicht gleich, müssen aber vom DBMS jeweils ausgeführt und neu zusammengestellt werden.

Ein Beispiel hierfür ist eine Anfrage auf eine Tabelle mit Maschinen. Ein Client benötigt für ein Simulationsmodell Daten Für die Maschinen M_1, M_2 und M_3. Zur gleichen Zeit trifft beim Query-Prozessor eine Anfrage für die Maschinen M_2, M_4 und M_5 ein. Bei einer konventionellen Ausführung würden zwei unabhängige Anfragen gestellt und die überlappenden Daten doppelt übertragen.

Bündelt man jedoch die Anfragen für jedes System innerhalb eines Thread-Pools, so können diese Zusammengefasst an die Datenbank weitergegeben werden. Dabei werden die Sichten mit den entsprechenden Parametern (z.B. M_1) an einen Thread übergeben und dieser wird einem übergeordneten Thread-Pool zugewiesen. Unter einem Thread versteht man in der Software-Architektur einen Ausführungsstrang innerhalb eines Prozesses, der parallel zu anderen Threads ausgeführt werden kann. Innerhalb des Pools können sich die Threads austauschen und somit eine einzige Anfrage mit fünf Maschinen als Parameter ausführen. Die Antwort der Datenbank wird wieder auf die einzelnen Threads aufgeteilt, welche anschließend ihren Auftraggeber damit versorgen.

Diese Technik ist besonders dann nützlich, wenn untergeordnete Abfragen ausgeführt werden müssen. Dies wäre z.B. der Fall, wenn für jede Maschine aus Datenbank A mehrere Betriebsmittel aus Datenbank B gelesen werden müssten und zusätzlich für jedes Betriebsmittel wieder die Maschinen aus Datenbank A, auf denen es eingesetzt werden kann.

Kapitel 4

Ein Simulationsdatenframework

In Abschnitt 2.3 sind die Anforderungen und Aufgaben bei der Simulation in der Teilefertigung beschrieben. Zur Erstellung von Simulationsmodellen in diesem Bereich benötigt man eine Großzahl von Daten, welche es erst ermöglichen die Realität in einem virtuellen Abbild darzustellen. Im Hinblick auf Simulationsanwendungen in einer bestehenden Fabrik ist es insbesondere wünschenswert den Zeitaufwand für die Abbildung der bereits existenten Anlagen, Maschinen und Abläufe möglichst gering zu halten. Soll ein neu zu beschaffender Durchlaufhärteofen in die bestehende Produktion integriert werden und dieser Ofen schon in der Planungsphase mittels digitaler Planungsmethoden und Simulation optimal ausgestaltet werden, so möchte man nicht einen Großteil der Zeit für das Erstellen eines Modells der gesamten Produktion aufwenden. Tatsache ist, dass die meisten Daten über die bestehende Produktion in diversen Systemen eines Industriebetriebs vorhanden sind oder durch Umwandlung oder Berechnung aus existenten Daten ermittelt werden können. Fakt ist jedoch auch, dass es sich dabei um eine gewaltige Menge an Daten handelt, welche von Hand kaum mehr ausgewertet werden und noch weniger auf formale Korrektheit und Validität geprüft werden können. Daher wird in den folgenden Abschnitten ein Rahmenwerk (engl. Framework) entwickelt, welches simulationsrelevante Daten rechnergestützt identifizieren, zusammen- und bereitstellen kann.

4.1 Wissenschaftliche Abgrenzung der Arbeit

Ein großer Nachteil der diskreten, ereignisgesteuerten Fabriksimulation ist die Tatsache, dass der größte Zeitanteil der Durchführung in der Eingangsdatenbeschaffung und Modellerstellung liegt [BCNN00, Ber00, LP98, UJ97, Try94]. Das Erstellen großer Modelle von ganzen Fabriken anhand traditioneller Methoden ist nicht nur zeit- und ressourcenaufwändig, auch der Aufwand die zugrundeliegenden Daten aktuell zu halten ist immens. Daher entwickelte sich in der Wissenschaft die Idee, die Simulationsdaten nicht mehr im Modell selbst, sondern in einer getrennten Datenbank zu halten, welche sich einfacher auf dem aktuellen Stand halten lässt. Leider erzeugt auch das Erstellen und die Pflege dieser noch großen Aufwand. Randell und Bolmjö ([RB01]) beschreiben eine Strategie, bei der der Aufwand für die datenbankgestützte Simulation besser verteilt werden kann, indem man die Datenbank gleichzeitig als Datengrundlage für andere Applikationen nutzt. Dabei reduziert sich der totale Aufwand für die Informationsverwaltung.

Verschiedene Autoren haben unterschiedliche Integrationsansätze zur datenbankgestützten Modellierung und Simulation beschrieben (vgl. Abb. 4.1). Ein Ansatz nutzt das Enterprise Modelling, um die darin enthaltenen Unternehmensdaten zur Generierung von Simulationsmodel-

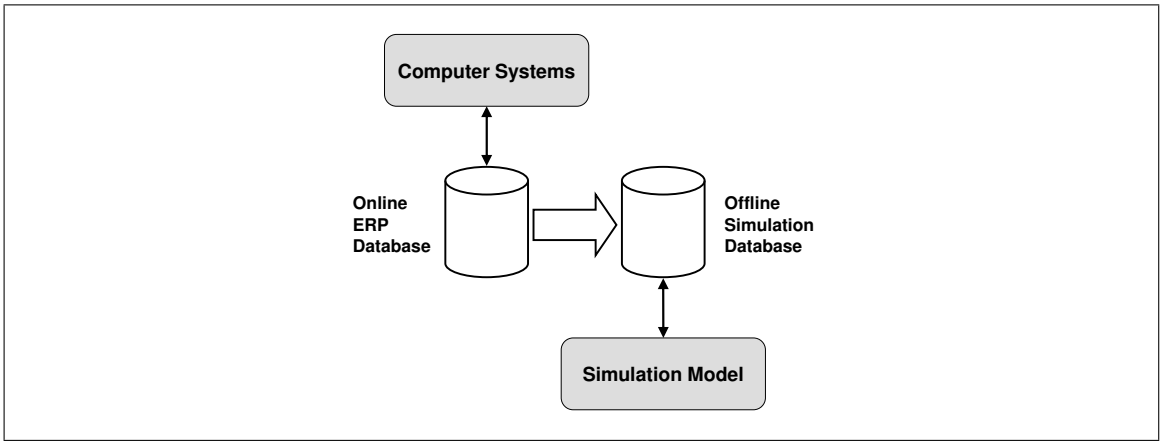


Abb. 4.1: Datenbankgestützte Simulation mit Simulationsdatenbank [RB01]

len wiederzuverwenden [DBE99, ZB99, DBE98, Hei97, SJ97, WHP97]. Bernard ([Ber00]) beschreibt einen Weg, Daten aus einem Prozessmodellierungstool wiederzuverwenden, um daraus Simulationsmodelle mit Hilfe des Austauschformats STEP[1] zu erstellen. Einen anderen, sehr interessanten Ansatz wählt die Technische Universität Ilmenau. Sie haben eine erweiterte Methode zur Erstellung von Simulationsmodellen entwickelt, welche auf mehrere, in einer Datenbank abgelegte Simulationskomponenten zurückgreift und daraus ein Gesamtmodell erstellt [GEP97, GEP98, EGP99]. Einen weiteren Ansatz liefert Johansson ([Joh01]). Er beschreibt eine Methode zur Speicherung, gemeinsamen Nutzung und Kommunikation von Daten aus Produktionssystemen mittels STEP.

Eine andere Forschungsrichtung beschreiben Love, Clarke und Gooden ([LCG87]). Sie haben schon sehr früh die Integration von Simulationsmodellen und MRP-Systemen gewagt. Die beiden Systeme wurden durch einen periodischen Austausch von Eingangs- und Ausgangsdaten miteinander gekoppelt. Ebenso beschreibt Reinhardt ([Rei85]) die Kopplung eines Simulators an eine Modellfabrik, wobei hier der Simulator zur Steuerung der echten Fertigung genutzt wird und somit als Leitstand dient. Eine weitere praktische Umsetzung dieser Integrationsthematik beschreibt Hotz ([Hot05]). Er stellt dar, wie einerseits Simulationslogik und Optimierungsverfahren aufbautechnisch getrennt werden und andererseits die Kopplung eines Leitstands der Elektrohängebahn im Getriebewerk der DaimlerChrysler AG ermöglicht wird.

Schließlich lässt sich noch das Gebiet der parametrisierbaren oder auch internetbasierten Simulation unterscheiden. Graupner, Richter und Sihn ([GRS02]) stellen die praktische Umsetzung eines über Intranet veränderbaren Simulationsmodells dar. Hierbei können sowohl generische Modelle, welche über Intranet bezüglich der Modellstruktur verändert werden können, als auch rein parametrisierbare Modelle, welche eine feste Struktur aufweisen, hinterlegt werden (vgl. Abb. 4.2). Weitere webbasierte Ansätze finden sich bei Fishwick ([Fis97]), Page ([Pag98]), Kuljis und Paul ([KP00]).

Im Rahmen dieser Arbeit soll der Umweg über proprietäre Methoden und eine offline Datenbank für Simulationsdaten umgangen werden. Es hat sich gezeigt, dass selbst bei einer

[1] Standard for the Exchange of Product model data

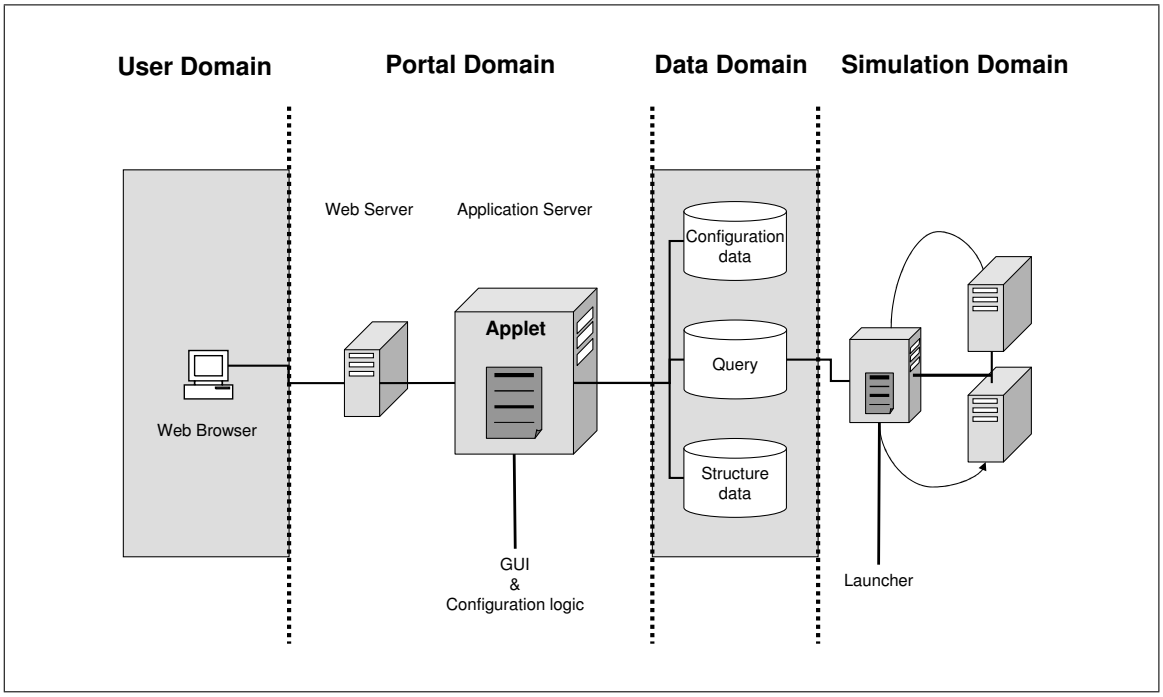


Abb. 4.2: Architektur einer webbasierten Simulationsanwendung [GRS02]

Trennung von Modell- und Datenstruktur noch ein großer Zeitaufwand zur Datenpflege und -zusammenführung anfällt. Desweiteren soll eine redundante Datenhaltung vermieden werden, um damit einer Dateninkonsistenz und fehlender Validität entgegenzuwirken. Das im Folgenden zu entwickelnde Framework zur Datenbereitstellung soll vorerst nur eine unidirektionale Datenrichtung implementieren. Dabei müssen aus den Produktions- und Dokumentationssystemen Daten für Simulationsmodelle gewonnen werden. Der bidirektionale Ansatz hat dabei eine niedrige Priorität. Es ist jedoch denkbar, dass planungstechnische Verbesserungen, welche innerhalb des Simulationsmodells erfolgen, später einmal direkt in die Produktivsysteme zurückgeschrieben werden. Um diesen Ansatz in die Realität umsetzen zu können, muss sich jedoch einiges in der Methodik und Technik der industriellen Planung ändern.

Ein weiteres Ziel dieser Arbeit ist die Schaffung der technischen Möglichkeit, beliebige Simulationsmodelle mit Realdaten zu versorgen, um damit die Verbindung von Materialflusssimulation und Realität, wie sie bei simulationsbasierten Frühwarnsystemen oder Online-Simulationen (vgl. [HTMS05]) vonnöten ist, zu schaffen. Beispielsweise kann dann das aktuelle Tagesproduktionsprogramm in den Simulator geladen werden, um am Modell zu verifizieren, ob dieses bei bestehenden Kapazitäten erfüllt werden kann, oder wie sich eventuelle Störungen auf den Produktionsablauf auswirken. Daher ist es erstrebenswert keine weitere Datenhaltungsschicht mehr zwischen den realen und den virtuellen Daten zu haben, wie sie in Abb. 4.1 dargestellt ist. Der Planungsprozess und auch der Planungsworkflow spielen bei der Anwendung von Simulation eine große Rolle, da diese am effizientesten ist, wenn sie über den gesamten Prozess hinweg eingesetzt wird. Daher soll im weiteren Verlauf der Arbeit nochmals genauer auf den Planungsablauf selbst eingegangen werden, um anschließend ein passendes Framework definieren zu können.

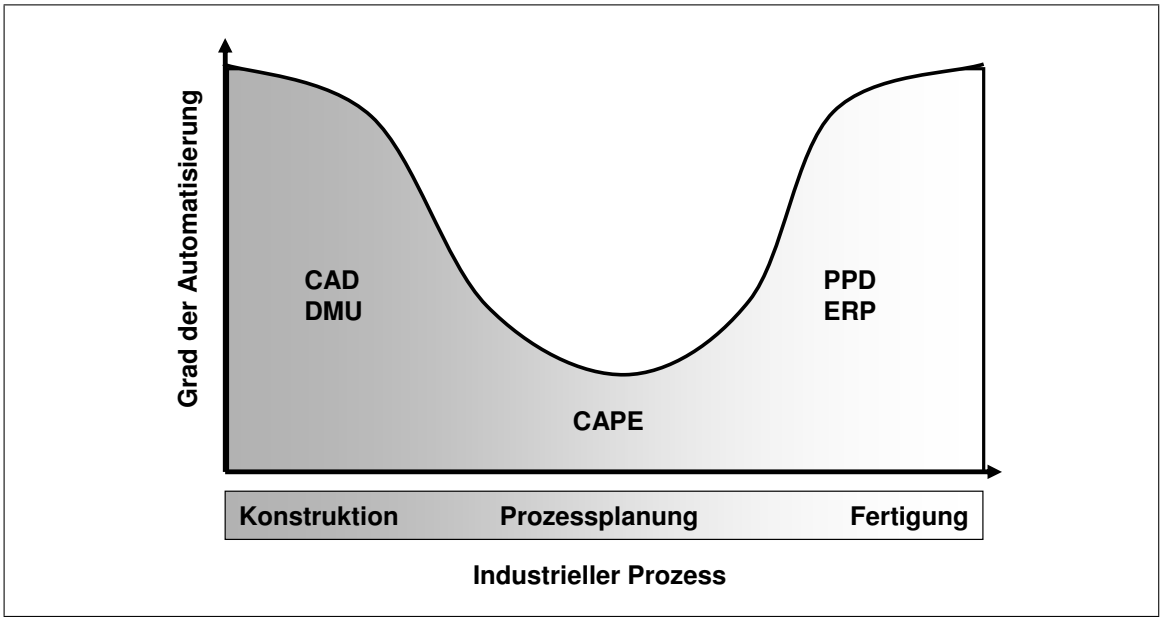


Abb. 4.3: IT-Durchdringung in den verschiedenen Planungsphasen [LGW99]

4.2 Darstellung des bisherigen Planungsprozesses

Bei der Beurteilung des konventionellen Planungsprozesses in der Automobilindustrie und insbesondere in der Aggregate- und Teilefertigung sind zwei wesentliche Bereiche identifizierbar, welche es zu verbessern gilt. Einerseits ist der IT-Durchdringungsgrad im Bereich der Produktionsplanung noch gering und andererseits gilt es den Ansatz des Simultaneous Engineering gezielt und sukzessive zu integrieren. In der Produktentwicklung werden durchgehend leistungsstarke Systeme, wie Catia und Pro Engineer, eingesetzt, um den Prozess im Bereich CAD und Digital Mockup (DMU) so effizient wie möglich zu gestalten. Gleiches gilt im Bereich der Serienproduktion, in der durchgängige Systeme für die Logistik und die Produktionssteuerung eingesetzt werden. Diese Enterprise Resource Planning (ERP) Systeme können heute einen Kundenauftrag direkt und optimiert in die Fertigung einlasten und dem Kunden sofort bestätigen, wann er mit seinem bestellten Produkt rechnen kann. In den letzten Jahren entwickelte Systematiken, wie „Just in Sequence“ (JIS) und „Just in Time“ (JIT) unterstützen diese schlanke und zeitlich abgestimmte Produktionseinlastung sehr. Lediglich auf dem Gebiet der Produktionsplanung sind solche IT-Systeme noch eher Insellösungen, als durchgängige und effiziente Hilfsmittel (siehe Abb. 4.3). Die durchgängige Vernetzung aller genannten Systeme von der Produktentwicklung bis hin zur Produktionssteuerung ist noch nicht vollzogen [Sch00b]. Systeme, die diese Lücke schließen sollen und die digitale Planung unterstützen, werden zum Teil auch unter dem Dachbegriff Computer Aided Production Engineering (CAPE) vereint [LGW99].

Viele Abläufe im Produktentstehungsprozess von der ersten Idee bis zur Umsetzung, also dem *Start of Production* (SOP), greifen noch nicht ineinander und laufen nacheinander ab (siehe Abb. 2.3 auf Seite 14). Das hat zur Folge, dass der Gesamtprozess länger dauert als nötig und auch größere Kosten verursacht, da mehr Kapazität für Abstimmungen und Änderungen be-

nötigt wird. Es lässt sich nicht vermeiden, dass die Entwicklung Produkte plant, welche bei der Umsetzung in ein Produktionskonzept zu aufwändig oder technologisch zu teuer sind. Die Fertigungsplanung muss wiederholt Änderungen mit der Konstruktion durchsprechen, wenn es darum geht, vorhandene Ressourcen in der Fabrik optimal auszulasten und stabile Prozesse zu schaffen. Eine konsequente Umsetzung des Simultaneous Engineering im Produktentstehungsprozess mit den zugehörigen Methoden Parallelisierung, Standardisierung und Integration kann den Wettbewerbsvorteil durch verkürzte Planungszeiträume, stärkere Kundenorientierung und geringeren Ressourcenbedarf enorm erhöhen [Wil93, VB05]. Es gilt daher, die Möglichkeiten des CAPE vollständig auszuschöpfen und in den Produktentstehungsprozess zu integrieren (vgl. Abb. 4.4).

Ein weiterer Aspekt, der den statischen Planungsprozess an seine Grenzen bringt, sind die stetig zunehmenden Planungsaufgaben in der Fertigung. Steigende Variantenzahlen, kürzere Produktlebenszyklen und eine härtere Wettbewerbssituation führen dazu, dass die Fertigung, meist bei gegebenen Ressourcen, kontinuierlich flexibler und kostengünstiger werden muss. Während in einer homogenen Linienfertigung mit wenigen Teilen theoretisch noch auf dem Papier geplant werden kann, so reichen die manuellen Hilfsmittel zur Planung eines dynamischen, flexiblen Fertigungsbereichs nicht mehr aus. Hier kommen die digitalen Planungsmethoden zum Einsatz, die es ermöglichen dynamische Abläufe bei vielen Varianten abzubilden. Insbesondere ist hier der Einsatz der Simulation gefragt, da diese das zeitgeraffte Durchspielen vieler verschiedener Alternativen ermöglicht.

Die Strukturen und Verantwortlichkeiten in mittelständischen und großen Industriebetrieben führen in der Planungsphase häufig zur Optimierung eines lokalen Bereichs, für welchen ein Planungsteam verantwortlich ist. Dabei werden Belange anderer Bereiche meist nicht berücksichtigt oder nur am Rande tangiert. Aus der Mathematik und dem Operations Research ist jedoch bekannt, dass die Optimierung auf lokal abgeschossenen Bereichen eines übergreifend zusammenhängenden Netzwerks meist nur zu mehreren lokalen Minima oder Maxima führt [NM02]. Ein Fertigungsplaner, welcher die Aufgabe hat, eine Wellenfertigung für Getriebewellen mit hohen Rüstzeiten der einzelnen Maschinen zu planen, wird versuchen die Summe aller Rüstzeiten in einem Betrachtungszeitraum zu minimieren, um Kosten zu sparen. Die Wellen kommen nach der Fertigung jedoch an ein flexibles Montageband, auf welchem verschiedenste Varianten verbaut werden, und welches von Kundenwünschen gesteuert wird. Der Planer für die Montage wird folglich seinen Optimierungsschwerpunkt auf ein möglichst flexibles Montagesystem ohne Rüstzeiten und mit minimalen Beständen legen. Beide Bereiche sind unmittelbar voneinander abhängig. Die einzelnen Optimierungsansätze sind jedoch stark konträr und lassen sich nicht miteinander vereinbaren. Würde jeder der beiden Planer sein lokales Optimum anstreben, so wären hohe Pufferbestände und lange Reaktionszeiten unvermeidbar. Es liegt nahe, dass das Gesamtsystem wesentlich effizienter geplant würde, wenn beide Planer mit Hilfe von rechnergestützten und eng verzahnten Planungssystemen zusammenarbeiten und Daten austauschen könnten. Gleiches gilt natürlich auch für die verschiedenen Planungsdisziplinen, wie Fertigungs-, Logistik- und Montageplanung. Auch in den Planungsrollen Vor-, Grob- und Feinplanung ist dieser Zustand wünschenswert. Ein wichtiger Aspekt für eine effektive Planungsarbeit ist dabei die Datenverfügbarkeit und der Datenaustausch, welche den Schwerpunkt dieser Arbeit darstellen.

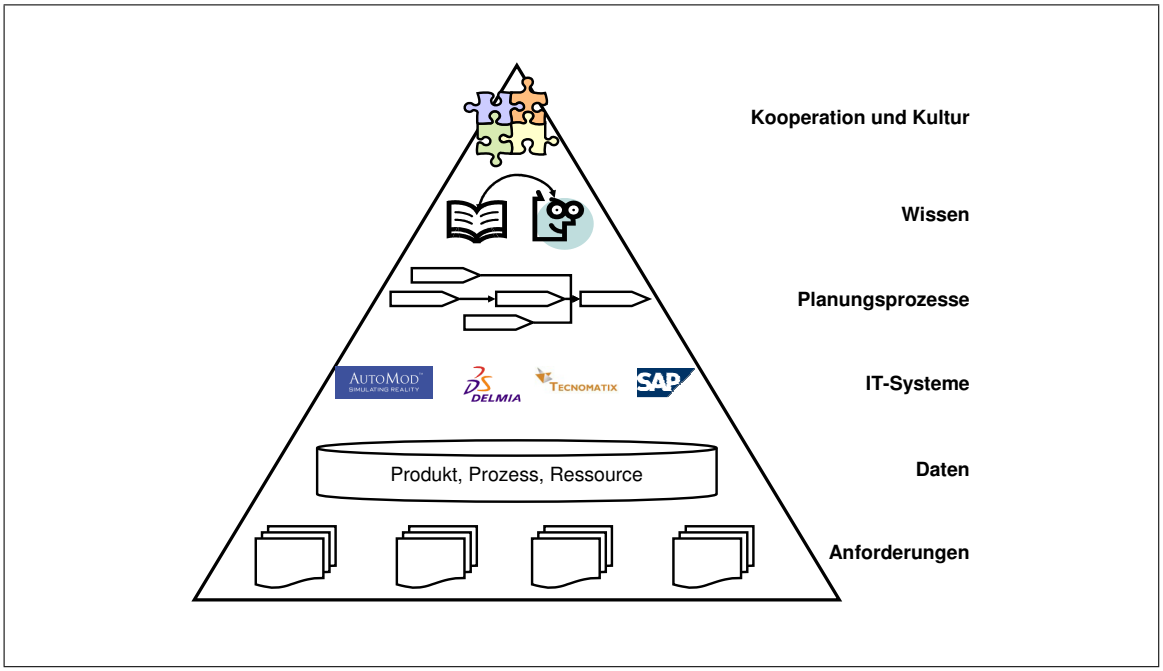


Abb. 4.4: Synchronisation der Prozessebenen zur nutzbringenden Anwendung digitaler Planungsmethoden [GB05]

Diverse Software-Integrationslösungen sind bereits auf dem Markt verfügbar (vgl. Abb. 4.5). Sie bieten Möglichkeiten, den oben genannten Mangel der IT-Durchdringung im Bereich der Produktionsplanung auszugleichen. Basis für die beiden marktführenden Produkte der Firmen Delmia und Tecnomatix ist eine zentrale Datenbasis in Form einer Datenbank zur gemeinsamen, gleichzeitigen und übergreifenden Nutzung von planungsrelevanten Daten. Sie sind heute bedingt durch immer leistungsfähigere Rechner und Server, auch bei der übergreifenden Nutzung von CAD-Daten, durchaus leistungsfähig und versprechen bei einer weiterhin starken Weiterentwickung ganz neue Möglichkeiten im Planungsprozess. Gleiches gilt auch für die Erreichung schnellerer Entwicklungszyklen.

Einen Mangel stellt jedoch noch immer die Anbindung der „alten“ IT-Welt dar. Bei der Rohbau- und Montageplanung im PKW-Bereich stellt dies keine große Relevanz dar, da mit dem Auslauf einer Baureihe in der Regel ein völlig neues Produkt quasi auf der grünen Wiese geplant wird. Mit dem heutigen Bestreben, bestehende Produktionsanlagen möglichst weiter zu verwenden, wird aber auch in diesem Sektor der Altwelt an Daten eine wachsende Bedeutung zukommen. In der Teilefertigung ist es schon lange Bestandteil, dass Produktionsanlagen für viele verschiedenartige Teile und Produkte verwendet werden und somit eine Neuplanung häufig auf der momentanen Serienproduktion aufbaut. Grund hierfür sind die Überschneidungen der Produkt-Lebenszyklen. Ältere Produkte laufen langsam aus und neue laufen an. Deshalb sind in der Teilefertigung die oben genannten Integrationssysteme noch eher spärlich im Einsatz, da die Integration der Daten der bestehenden Produktion in die digitalen Planungssysteme häufig eine Herausforderung darstellt. Bedingt durch die hohen Anschaffungs- und Folgekosten für Schnittstellenentwicklungen, Softwareanpassungen, Mängel bei der Datenintegrität und die

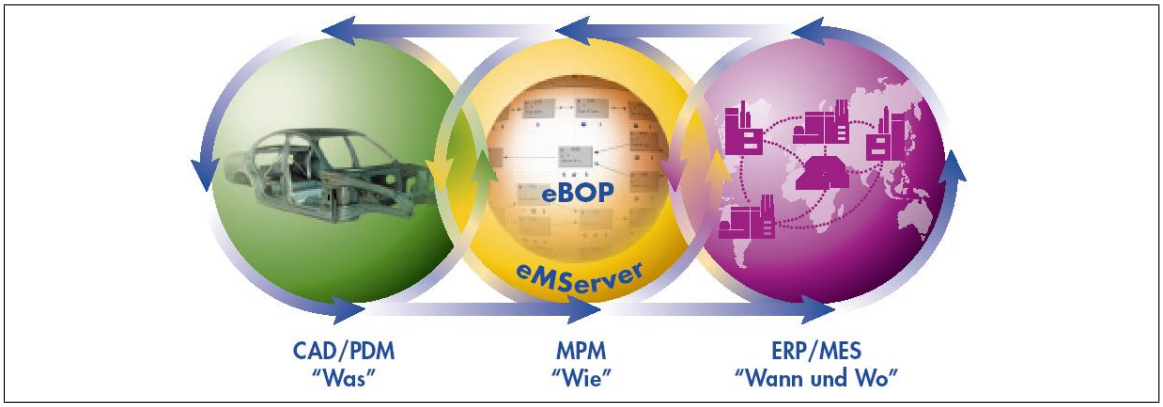


Abb. 4.5: Tecnomatix Integrationslösung für die digitale Produktionsplanung

Sicherstellung der Rückversorgung von Daten in Altsysteme, erreichen diese Systeme in der Aggregateplanung noch nicht den gewünschten Implementierungsgrad.

Insbesondere die Integration der bestehenden Systemwelt und die Rückversorgung neuer Plandaten in die serienrelevanten Steuerungssysteme, wie ERP-, MRP-, Kalkulations- und Lohnabrechnungsssysteme, stellt eine zentrale Rolle für die erfolgreiche Umstrukturierung des alten Planungsprozesses dar. Bei dem im Rahmen dieser Arbeit zu entwickelnden Datenframework soll dabei ein besonderes Augenmerk auf die Echtzeit- oder Online-Kopplung dieser Daten gelegt werden. Die Methode soll eine redundante Datenhaltung vermeiden und sich modular auf verschiedenste Anwendungen und Systeme adaptieren lassen, ohne dass dabei großer Aufwand für Anpassungen und Erweiterungen entsteht. Der integrierte, neue Planungsprozess soll den in Abb. 4.3 dargestellten Mangel an effektiven Systemen in der Produktionsplanung beheben und die Zusammenhänge zwischen Entwicklung und Serienfertigung harmonisch zusammenführen (vgl. Abb. 4.6). Bevor jedoch die Entwicklungsschritte näher erläutert werden, soll vorab erst einmal identifiziert werden, welches Grundgerüst an Daten notwendig ist, um eine erste Integrationsstufe zu erreichen.

4.3 Identifikation simulationsrelevanter Daten

Die Menge, Art und Anzahl der benötigten Daten zur Durchführung von Simulationsstudien variiert stark mit der Zielsetzung, dem Detaillierungsgrad und der Methode der durchzuführenden Simulationsstudie. Dabei sind die möglichen Einsatzgebiete der Rechnersimulation sehr vielfältig und können sich auch überlappen oder ineinander greifen. Hinzukommt, dass Produktionssysteme, Prozesse und Datenmengen immer weiter ins Komplexe wachsen. Produktentwicklung, Produktionsengineering und -management involvieren oftmals die Betrachtung vieler voneinander abhängiger Variablen [MHZL03]. Bei einer groben Untergliederung lassen sich die folgenden Einsatzgebiete und die zugehörigen, notwendigen Daten (vgl. Abb. 4.8) identifizieren:

- Logistikabläufe

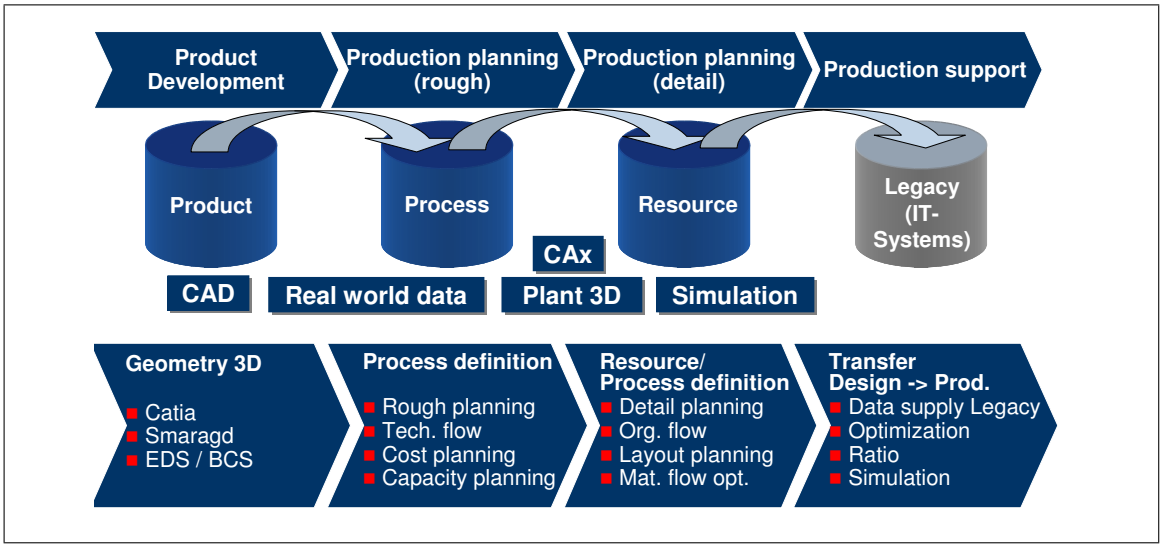


Abb. 4.6: Darstellung eines integrierten Planungsprozesses

- Materialfluss
- Werkereinsatzstrategien
- Ergonomie
- Bewegungssimulation und Robotik

Zu den Logistikabläufen zählen jegliche Abläufe von Materialtransport, -lagerung und auch Teile der Fabriksteuerung (vgl. Abb. 4.7). Dabei kann der Materialtransport durch Gabelstapler, Menschen und Fördertechnik erfolgen. Zur Lagerung zählen Pufferplätze, Supermärkte, Hochregallager und auch neuere Konzepte der Einbeziehung von Transportwegen, wie Schiff und LKW, welche als zusätzliches, externes Lager gesehen werden können. Logistikkonzepte wie JIT und JIS (vgl. [KK00]) können mit Hilfe der Simulation bzgl. Lieferzeitpunkten, -mengen, Terminvorschau und Transport ausgeplant und optimiert werden. Auch die Bestell- und Anliefermodalitäten für Zulieferteile, welche am Montageband bereitgestellt werden, sind von Bedeutung. In moderneren Montageanlagen bewegt sich das zu montierende Produkt oftmals mit konstanter Geschwindigkeit auf einem Band weiter, was zu einer so genannten „Drift“ führt. Das kann zur Folge haben, dass Bereitstellungsplätze flexibel gestaltet sein müssen und das Logistikkonzept dementsprechend aufwändig zu planen ist. Auch das Produktionsprogramm in Bezug auf Baumusterauflösung, Absatzplanung, Stückzahlen und Variantenmix ist für die Simulation unverzichtbar.

Klassisches Gebiet der Simulation ist die Dimensionierung des Materialflusses. Hierzu zählen die Abläufe der unterschiedlichen Fertigungsschritte innerhalb der Fabrik und die Bewertung verschiedener Fertigungskonzepte bzgl. Durchlaufzeiten, Auslastungen, etc.. Der Materialfluss und die Logistikabläufe werden häufig gemeinsam betrachtet, da zu einer stabilen Prozesskette auch passend dimensionierte Supermärkte und Pufferflächen gehören. Auch die Werkereinsatzstrategien werden durch den Materialfluss und das Fertigungskonzept beeinflusst. Themen wie

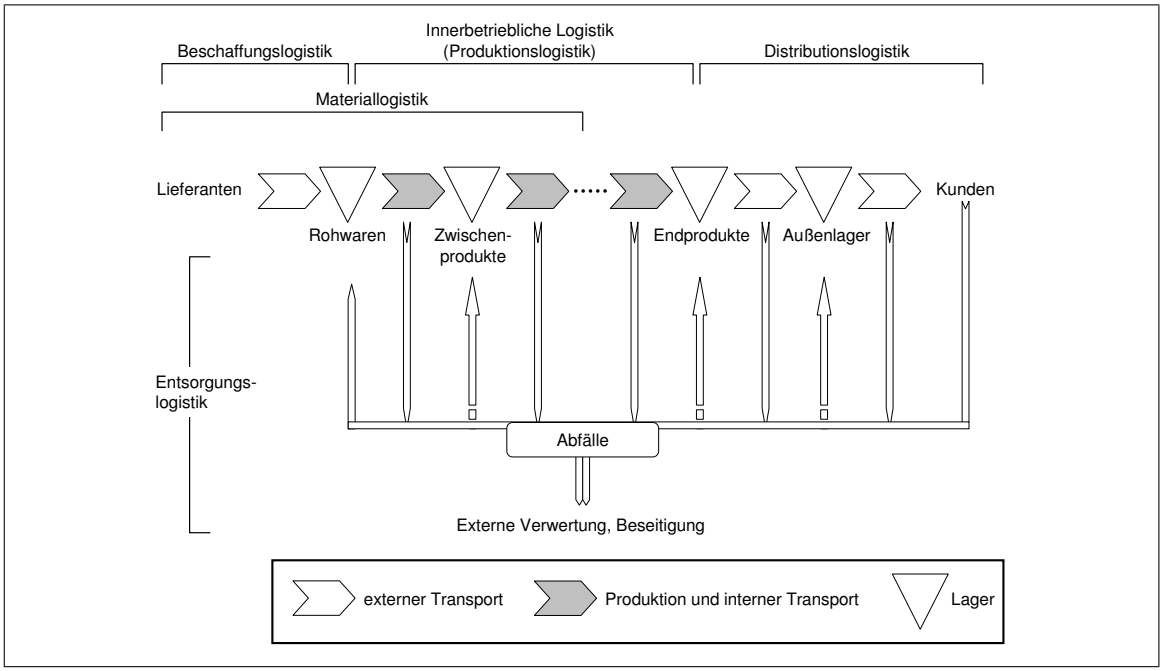


Abb. 4.7: Allgemeine Darstellung einer Logistikkette [Arn04]

Wegstreckenminimierung, Schichtmodelle und automatisierte Fertigungsprozesse spielen hierbei eine Rolle. Zu den industriellen Datenverarbeitungssystemen, welche simulationsrelevante Informationen bereithalten, gehören unter anderem CAD-, Dokumentations-, Instandhaltungs- und Leitstandsysteme. Wichtige Bestandteile stellen Informationen über Fertigungszeiten, Wegstrecken, technische Verfügbarkeiten, Fertigungskosten, etc. dar.

Die Ergonomiebetrachtungen werden häufig für einzelne Arbeitsplätze und -stationen durchgeführt. Jedoch können auch bei der Ergonomiesimulation mehrere Abläufe miteinander verbunden werden und als Ganzes betrachtet werden. Diese Disziplin beinhaltet auch die Themen der Bewegungssimulation und Robotik. Es gilt nicht nur menschliche Arbeitsplätze ergonomisch optimal auszulegen, sondern auch automatisierte Abläufe (z.B. Roboter) möglichst effizient zu gestalten. Roboter können offline im dreidimensionalen Modell programmiert und auf andere Abläufe ausgetaktet werden. Systeme, die Daten über die genannten Anwendungen liefern, sind z.B. Zeitberechnungs- und Dokumentationssysteme (MTM[2], NC-Steuerungen[3] und Leitstandsysteme).

Für jede dieser Simulationsarten sind unterschiedliche Daten für die Modellierung und die Durchführung von Simulationsstudien notwendig. In Abschnitt 2.3 wurde bereits erwähnt, dass in der Teile- und Aggregatefertigung größtenteils Kombinationen aus Logistik-, Materialfluss- und Werkereinsatzsimulationen durchgeführt werden, um Produktionskonzepte und Fertigungsabläufe zu bewerten. Auch die hierfür benötigten Daten sind in Abb. 4.8 bereits dargestellt. Aufgrund des großen Umfangs der möglichen Einsetzbarkeit muss im Rahmen der Entwick-

[2]Methods Time Measurement
[3]Numeric Control

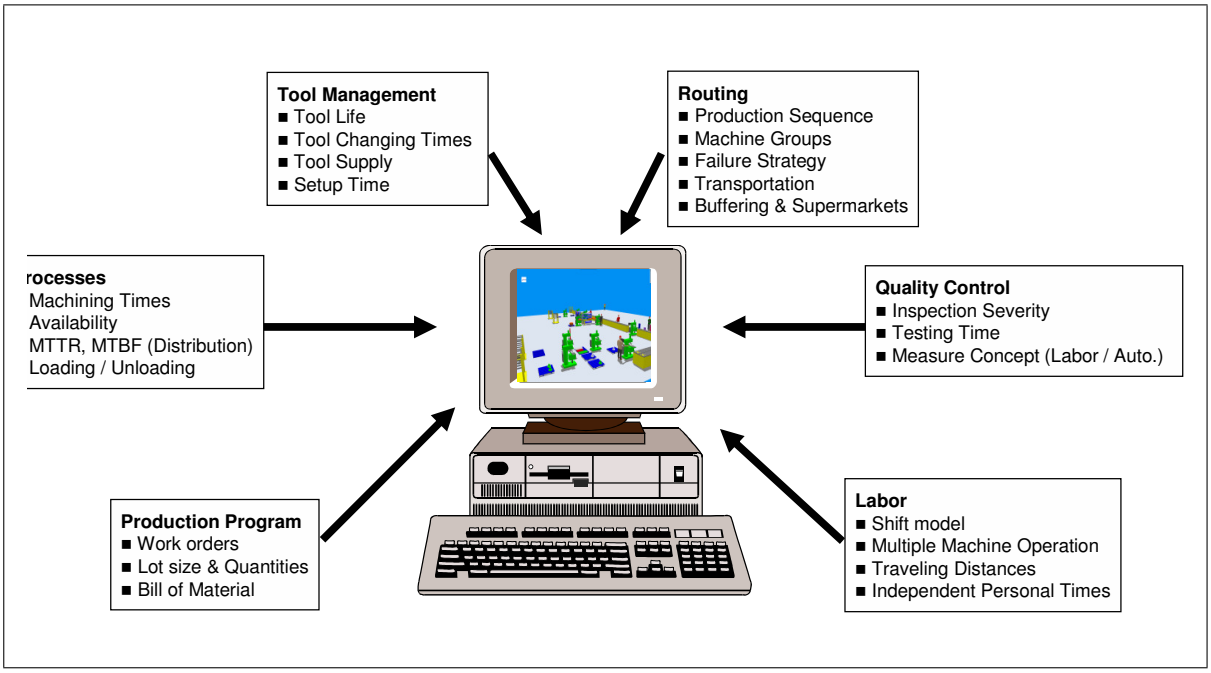


Abb. 4.8: Simulationsrelevante Daten aus industriellen DV-Systemen [RVJ03]

lung einer Simulationsdatenstruktur jedoch ein hohes Maß an Flexibilität und Erweiterbarkeit berücksichtigt werden.

In Kapitel 3 wurde die Entwicklung der industriellen Datenverarbeitung und die Problematik der heutigen, verteilten und systemübergreifenden Datenhaltung erläutert. Den in diesem Kapitel erwähnten, für die Simulation relevanten Systemen unterliegen ausschließlich Datenbanksysteme unterschiedlicher Hersteller, Versionen und Aufbaustrukturen. Es soll daher der Ansatz von Webtechnologien gewählt werden, um eine modulare, skalierbare Anbindung von Produktionsdatensysteme an die virtuelle Realität umzusetzen.

4.4 Entwicklung einer Simulationsdatenstruktur

Es wurde bereits dargestellt, dass die meisten simulationsrelevanten Daten in Industriebetrieben aus Datenbanksystemen unterschiedlichster Architektur stammen und dort zu Planungs- und Dokumentationszwecken bereitgestellt werden. Große Mengen an produktions- und steuerungsrelevanten Daten durchlaufen in der Industrie Client- und Server-Systeme verschiedener Hersteller und Architekturen. Die Herausforderung liegt in einer konsistenten Darstellung und einheitlichen Datenformaten [MHZL03]. Bei der Auswahl eines geeigneten Verfahrens zur Generierung einer Simulationsdatenstruktur muss beachtet werden, dass die Struktur der Rohdaten zumeist von relationalen Systemen (Datenbanken) bestimmt wird, und dass sich dieser Ursprung in der Ausprägung der Daten widerspiegelt. Im relationalen Modell werden Daten in verschiedenen Tabellen, entweder lokal oder verteilt, gehalten und sind über definierte Schlüsselfelder miteinander Verknüpft. Eine Simulationsdatenstruktur muss folglich in der Lage sein, sowohl

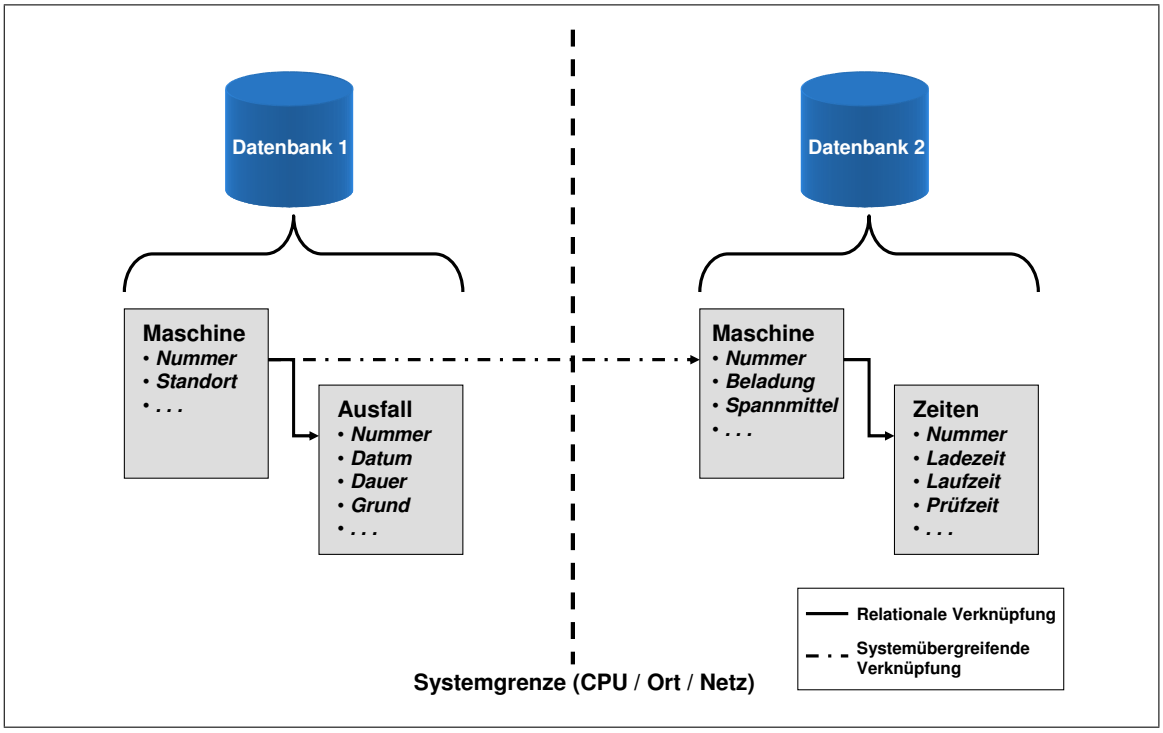


Abb. 4.9: Systemübergreifende Zusammengehörigkeit von simulationsrelevanten Daten

diese genannten Strukturen, als auch die flexible und erweiterbare Verknüpfung mehrerer dieser Datenbanken abzubilden (siehe Abb. 4.9). Schwierigkeiten, wie die Tatsache, dass Schlüsselfelder zwischen verschiedenen Datenbanken historisch bedingt in der Syntax nicht immer identisch sind, müssen dabei berücksichtigt werden. So kann es vorkommen, dass Inventarnummern in einem System mit und in einem anderen ohne Leerzeichen auftreten. Außerdem können Schlüsselfelder unterschiedliche Datentypen, wie String (Text) und Integer (Ganzzahl) aufweisen.

4.4.1 Aufgabenstellung

Bei der Wahl eines geeigneten Verfahrens soll die Flexibilität und Wiederverwendbarkeit im Vordergrund stehen. Bezüglich der Technik soll das Verfahren auf zeitgemäßen Techniken aufbauen, welche die genannten Anforderungen erfüllen, und welche direkt in eine vernetzte Unternehmensdatenstruktur implementiert werden können. Während heutige Datenbanksysteme im Bereich der übergreifenden Datennutzung oftmals eingeschränkt sind und feste Systemgrenzen aufweisen, bieten Webtechnologien, wie Java, XML und Webservices, Methoden und Techniken zur skalierbaren und flexiblen Vernetzung unterschiedlichster Daten und Systeme. Gerade heute werden in Unternehmen mehr und mehr kosten- und verwaltungsintensive Schnittstellen zwischen Softwareprodukten geschaffen, ohne den Hintergrund der Wiederverwendbarkeit und Mehrfachanwendung zu berücksichtigen. Daher sind die Webtechnologien insbesondere auch bezüglich Kostenaspekten sehr vielversprechend.

Viele, mittlerweile alltäglichen Aufgaben, wie Bankgeschäfte, Reisebuchungen, Warenbestellungen und das Abrufen von Wetterinformationen können mit Hilfe der genannten Webtechni-

ken heute von jedem beliebigen Ort auf der Welt online erledigt werden. Der Vernetzung von Daten und Anwendungen sind keine unüberwindbaren Grenzen mehr gesetzt. Der Stand der Technik auf diesem Gebiet und die bereits vorhandenen Anwendungen lassen keinen Zweifel, dass auch im Bereich der Integration von Unternehmensdaten in die Rechnersimulation, die Webtechnologien Möglichkeiten bieten. Die Zielsetzung dieser Aufgabe kann mit deren Einsatz effizient und kostengünstig ermöglicht werden. Insbesondere im Bereich der Materialflusssimulation wird auch heute noch mit Datenexporten in Form von Textdateien und Datenabzügen in speziell für den Anwendungsfall geschaffene Datenbanken gearbeitet. Dieser Ansatz ist jedoch zeitintensiv und prädestiniert für Dateninkonsistenzen.

Der Trend bei der Weiterentwicklung der kommerziellen Simulatoren geht immer mehr in Richtung der flexiblen Datenintegration. So unterstützt der Simulator Quest der Firma Delmia schon heute die Verknüpfung von Simulationsentitäten mit der auch aus diesem Softwarehaus stammenden Planungsdatenbank (PPR-Hub[4]). Auch der Simulator Automod der Firma Brooks Automation bietet in den kommenden Versionen eine direkte Schnittstelle zu ODBC[5]-Datenbanken an und kann schon heute Dateien importieren. Bei vielen Simulationssystemen sind bereits XML-Integrationen implementiert, oder zumindest für künftige Versionen angedacht. Die Entwicklung eines Simulationsdatenframeworks in XML bietet folglich für die Zukunft ganz neue Möglichkeiten bei der Modellerstellung. Es wäre denkbar, einen momentanen Zustand der Fabrik in Echtzeit in den Simulator zu übertragen, um dann von der Realität abgekoppelt verschiedene Szenarien (z.B. Ausfall- oder Werkereinsatzstrategie) durchzuspielen. Diese Idee beschreibt Reinhardt ([Rei85]) schon im Jahr 1985. Heute ist die Technik auf einem Stand um diese Ideen effizient umsetzen zu können und es sind noch viele weitere Anwendungsmöglichkeiten denkbar.

In einem ersten Schritt soll eine Systematik entwickelt werden, welche eine standardisierte Darstellung von simulationsrelevanten Daten ermöglicht. Diese soll flexibel anwendbar sein und bei Bedarf automatisiert Daten aus Planungs- und Dokumentationssystemen liefern. Auch die dynamische Komponente bei der Bereitstellung der Daten spielt dabei eine wichtige Rolle, da es in den wenigsten Fällen vonnöten ist alle Daten eines bestimmten Bereichs komplett zur Verfügung zu stellen. Dadurch wird die Möglichkeit geschaffen, dass nicht nur Simulationssysteme die Daten nutzen können, sondern dass diese auch in beliebigen anderen Anwendungen, wie Informations-, Vorplanungs- und Berechnunssystemen genutzt werden können. Dabei gilt es jedoch stets zu beachten, dass verschiedene Sichtweisen auf bestehenden Daten existieren. So ist beispielsweise eine Produktionsfläche aus Sicht des Controllings weniger detailliert, als aus Sicht der Produktionsplanung, welche Logistik-, Produktions, Versorgungsfläche, etc. unterscheidet. Im weiteren Verlauf der Arbeit muss folglich ein besonderes Augenmerk auf den notwendigen Detaillierungsgrad der Daten gerichtet werden, um dementsprechend das System als Datenquelle zu identifizieren, welches diesem am ehesten gerecht wird.

Bei der Datenzusammenführung, -darstellung und -übermittlung trifft man immer häufiger auf das XML-Format. Dieses soll im folgenden Abschnitt näher erläutert werden und in Bezug auf die Anwendbarkeit zur Darstellung von simulationsrelevanten Daten geprüft werden.

[4] Produkt-Prozess-Ressourcen-Hub (siehe auch http://www.delmia.com)
[5] Open Database Connectivity

4.4.2 Die Extended Markup Language XML

Die Themenbereiche der verteilten Datenbanksysteme und Datenbank-Cluster werden immer stärker in Verbindung miteinander gesehen. XML für den Austausch von Dokumenten und Informationen ist seit der Standardisierung 1998 vom World Wide Web Consortium (W3C) eines der wichtigsten Themen im Bereich der Informationssysteme und Datenbanken geworden [W3C98]. Eine derartige Entwicklung ist in sich schon überraschend, da XML eigentlich nichts weiteres ist, als ein selbstbeschreibendes Datenformat, welches Schema und Ausprägung in einer Datenstruktur verbindet und daher in der traditionellen Datenbankdenkweise aus Optimierungs- und Integritätsgründen eher vermieden wurde [Gra03].

Es gibt mehrere Gründe für den Erfolg und die rasche Entwicklung im Bereich der Anwendung von XML. Der wichtigste ist das der XML unterliegende, sehr flexible und teilstrukturierte Datenmodell. So kann mit XML der gesamte Bereich an Informationsausprägungen, von unstrukturierten Daten (z.B. Klartextdokumente), bis hin zu stark strukturierten Daten (z.B. Relationen), abgedeckt werden [ABS00]. Ein weiterer wichtiger Punkt ist die selbstbeschreibende Darstellung von XML. Daten und Schema werden zu einer einzigen Repräsentation, der sogenannten XML-Dokumente, zusammengeführt. Daher sind sie leicht von den unterschiedlichsten Anwendungen zu interpretieren. Letztlich baut XML auf Internet-Standards, wie HTTP[6], zur Datenübertragung auf. Daraus folgt eine starke Vereinfachung beim Austausch von Daten über das Internet oder zwischen verschiedenen Datenhaltungssystemen [Gra03].

Der Begriff extensible (erweiterbar) in XML beruht auf der Tatsache, dass XML kein festes Format aufweist, wie das unter anderem bei HTML[7], einer Sprache mit einer einzigen, vordefinierten Ausprägung, der Fall ist. XML ist eine Meta-Sprache und daher in der Lage, andere Sprachen zu beschreiben. Es können angepasste Beschreibungsmöglichkeiten generiert werden, welche es ermöglichen, jedes beliebige Dokument darzustellen. Grundlage hierfür ist der Aufbau von XML auf dem internationalen Standard für Meta-Sprachen (SGML[8] - ISO 8879)[ISO86].

Ebenso wie bei SGML, besteht ein XML-Dokument aus einem Dokumenttext, welcher sowohl aus *markup* (Beschreibung) als auch aus *content* (Inhalt) zusammengesetzt ist. Abb. 4.10 zeigt eine vereinfachte Darstellung der simulationsrelevanten Daten einer Maschine. Das Markup definiert sogenannte *elements* (Elemente). So stellt `<Bezeichung>` und `</Bezeichung>` das Markup für die Bezeichung einer Maschine dar. Dabei nennt man den Namen eines Elements auch häufig sein *label* (Kennzeichen). Der zugehörige Inhalt des Elements ist ein Maschinenname, wie „EMAG VCF100“. Das Wurzelelement - in diesem Fall `Modell` - nennt man auch das *Document element*.

Es gilt zu beachten, dass der Inhalt eines Elements rekursiv weitere Elemente und Markup enthalten kann. So könnte unterhalb der Struktur einer Maschine ein Element vom Typ *Referenz auf eine andere Maschine* stehen. Damit könnte die Zugehörigkeit zu einem Maschinenverbund ausgedrückt werden. Analog könnten dann auch bei diesen wieder Referenzen auftauchen und es bildet sich eine Rekursion. Man sieht an diesem Beispiel sehr gut, dass als weiteres wichtiges

[6] Hypertext Transfer Protocol (siehe auch http://www.w3.org/Protocols RFC 2616)
[7] Hypertext Markup Language (siehe auch http://www.w3.org/MarkUp ISO 8879 und RFC 1866
[8] Standard Generalized Markup Language

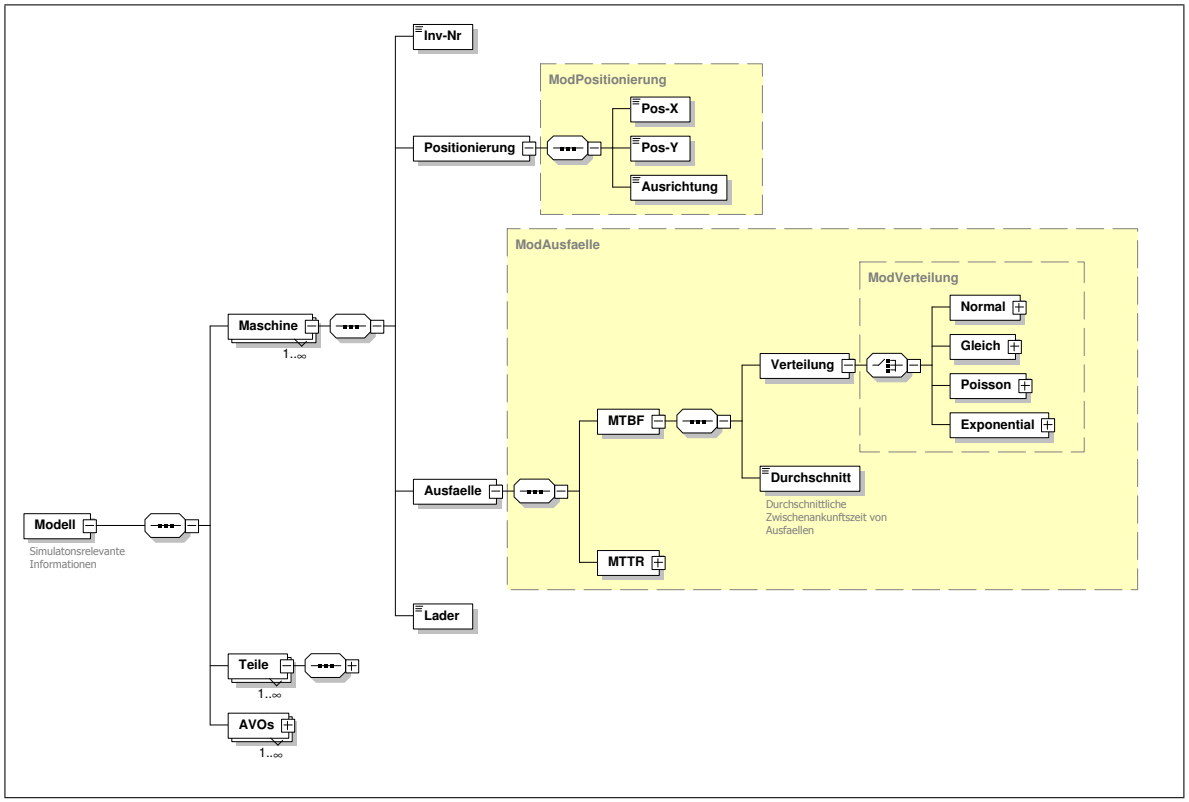


Abb. 4.10: Schematische Darstellung der XML-Datenstruktur für Simulationsmodelle

Charakteristikum eines teilstrukturierten Datenmodells die Tatsache auftritt, dass es keine klare Aufteilung zwischen Daten und Datenstruktur bzw. Schema gibt. Anstelle dessen wird sowohl Datum als auch Datenstruktur innerhalb eines einzigen XML-Dokuments wiedergegeben. Dieses Merkmal bildet einen grundlegenden Unterschied zu strikt strukturierten Datenmodellen in einem relationalen Modell, wie man es von Datenbanksystemen kennt. Das erklärt einerseits die hohe Flexibilität von XML und andererseits macht es eine effiziente Datenverarbeitung zu einer größeren Herausforderung, als bei strikt strukturierten Datenmodellen.

XML-Dokumente können zur vereinfachten Veranschaulichung in einer Baumstruktur dargestellt werden. Der hierarchische Aufbau solcher Dokumente beruht auf der Empfehlung des W3C, wie sie in der Beschreibung von XPath definiert ist [W3C99]. In Abb. 4.10 wird eine solche Baumstruktur repräsentiert. Es ist ersichtlich, dass textueller Inhalt stets an den Blättern des Baums auftreten. Man nennt ein XML-Dokument auch *well-formed* (wohlgeformt), wenn es ein Wurzelelement, wie in Abbildung 4.10 die `Maschine`, aufweist und alle anderen Elemente hierarchisch unter diesem Element verschachtelt sind. Zusätzlich zur Wohlgeformtheit sieht XML mehrere Möglichkeiten zur Einschränkung von Elementtypen, wie der Struktur von Elementen und deren hierarchische Verschachtelung innerhalb des Dokuments, vor. Diese Einschränkungen werden entweder in einem *XML-Schema* [W3C04], oder einer *Document Type Definition* (DTD) [W3C98] definiert. Sind XML-Dokumente wohlgeformt und konform zu ihrer DTD bzw. ihrem Schema, so bezeichnet man sie auch als *valid* (valide). Es besteht folglich eine Möglichkeit, simulationsrelevante Daten anhand eines Schemas zu validieren.

Ein DTD oder ein Schema, verglichen zu den relationalen Schemadefinitionen in Abschnitt 3.1.4, ist das Faktum, dass sich auch weniger strukturierte Daten abdecken lassen, als dies beim relationalen Modell möglich ist. In dem Beispiel aus Abbildung 4.10 ist diese Tatsache bei der MTTR und der MTBF exemplarisch dargestellt. Sowohl das Element `MTBF`, als auch das Element `MTTR` können auf zwei verschiedene Arten instanziiert werden: als Subelement `Verteilung`, welches wiederum aus Subelementen besteht, oder mit einem konstanten Wert (z.B. 5.0h) ohne weitere Subelemente. Diese Flexibilität ist mit einer der Gründe, warum sich XML sehr gut zur Repräsentation von teilstrukturierten Daten eignet [ABS00, AQM+97]. Für ein XML-Dokument ist eine DTD oder ein Schema nicht zwangsläufig notwendig. Ist keine von beiden vorhanden, so ist auch die Struktur nicht a priori bekannt. In den meisten Fällen ist es jedoch vonnöten, a posteriori Kenntnis über die Struktur zu haben, da insbesondere bei der Zusammenführung mehrerer Dokumente ansonsten eine Verarbeitung kaum möglich wäre.

4.4.3 Standardisierte Modellbeschreibung

Die Idee, ein neutrales Beschreibungsformat für Simulationssysteme zu entwickeln, ist nicht neu. Insbesondere im CAD-Bereich wurden in der Vergangenheit schon große Anstrengungen unternommen, standardisierte Austauschformate zu definieren, welche heute produktiv eingesetzt werden. Hierzu zählen z.B. CAD-Austauschformate wie DXF, IGES und STEP [MHZL03]. Aus der Sicht der Softwarehersteller stellt sich dabei zuerst die Frage, was standardisiert werden soll. Bezieht sich die Standardisierung auf die Benutzeroberfläche (GUI[9], etc.), die Modellierungsmethode (objektorientiert, etc.), die Ausgabeformate (Excel, etc.) oder die Interoperabilität (HLA, etc.). Große Bedeutung wird der Vereinheitlichung von Modelldateien, also der Speicherform von Simulationsmodellen zugemessen.

Im Bereich der Modellbeschreibung wurden bereits große Fortschritte mit den Spezifikationen des „Simulation Data Exchange“ (SDX) erzielt. Dieses Format wurde ursprünglich als allgemeingültiges Datenformat zur Generierung von diskreten, ereignisgesteuerten Simulationsmodellen und 3D-Modellanimationen direkt aus CAD-Systemen entwickelt. In SDX sind Dateiattribute und Informations- und Objekteigenschaften zwischen `[Begin]` und `[End]` Markierungen definiert. Diese werden in einer Textdatei gespeichert und sollen die einfache Lesbarkeit und Verständlichkeit gewährleisten (vgl. auch [SM01]). SDX unterstützt den Ansatz der Integration von Simulationssystemen in Layoutplanungsanwendungen. Es ist u.a. in FactoryCAD der Firma Unigraphics Solutions (UGS) integriert. Fabrikobjekte werden mit zusätzlichen Daten, wie MTBF, Kapazitäten und Bearbeitungszeiten hinterlegt und können über diesen Standard an ein Simulationssystem übergeben werden. Damit sollen Experimente direkt im Layoutplanungssystem durchführbar sein. Der Ablauf stellt sich folgendermaßen dar [Hat01]:

- Ein 3D-Modell des Layouts wird generiert.
- Simulationsrelevante Daten werden zu den Fabrikobjekten ergänzt.
- Mit dem SDX Route Editor werden die Materialflüsse deklariert.
- Eine formatierte Textdatei mit Ablauf- und Layoutdaten wird erzeugt.

[9] Graphical user interface

- Die SDX Datei wird in ein Simulationssystem überführt.
- Der Durchsatz der geplanten Fabrik wird in einem Simulator ausgewertet.

SDX ist ein guter, erster Entwurf für ein neutrales Austauschformat [MHZL03]. Jedoch bestehen auch Einschränkungen und Kompatibilitätsprobleme. Der Ansatz ist besonders Leistungsstark in Bereichen mit einer Vielzahl automatisierter Materialflusstechnik, wie Transportbändern, fahrerlosen Transportsystemen (FTS), Gabelstaplern und Kränen. Außerdem muss der Anwender Erfahrung in der Simulation, Statistik und Warteschlangentheorie haben [ÜSC+99]. Einen weiteren, vielversprechenderen Ansatz stellt das XML-Format dar, welches sich bereits in einigen Anwendungsfeldern rund um die Simulation hervorhebt und erste Erfolge ermöglicht [MHZL03, Ban00].

Die größte Barriere auf dem Weg zu einem standardisierten Modellierungsformat stellen jedoch die Hersteller der Software selbst dar. Es stellt sich die Frage, worin das Interesse eines etablierten Softwareproduzenten liegen könnte, eine übergreifende Nutzung von Simulationsmodellen zu ermöglichen. Zur Beantwortung dieser Frage muss man die folgenden Gesichtspunkte seitens der Anbieter berücksichtigen [MHZL03]:

- Wie würde ein Standard den mittel- und langfristigen Absatz beeinflussen?
- Wie hoch ist das wirtschaftliche Risiko, wenn der Standard nicht unterstützt wird?
- Wie wirkt sich der Standard auf den Wettbewerbsvorteil aus?

Es lässt sich vermuten, dass die Auswirkung auf den Absatz der wichtigste Entscheidungsfaktor ist, da sich dieser direkt in der Kostenrechnung und Bilanz auswirkt. Gesteigerte Absätze sind proportional zu einem höheren Marktanteil oder einem größeren Absatzmarkt. Ein größerer Marktanteil ist das Resultat aus der Bereitschaft von Unternehmen, das Produkt zu kaufen, da es die gemeinsame Nutzung von Daten systemübergreifend und den Import von Modellen, die mit anderen Produkten erzeugt wurden, erlaubt. Der Markt selbst wächst, wenn Unternehmen wahrnehmen, dass Simulation einfach in die Systemlandschaft integriert werden kann und eine gemeinsame Datennutzung unterstützt wird.

4.5 Entwicklung eines Datenframeworks

In der Softwareentwicklung von Client-Server-Anwendungen, wie es hier der Fall ist, unterscheidet man zwischen zwei wichtigen Architekturen. Einerseits gibt es die *two-tier architecture* (zweischichtige Architektur) und andererseits die *three-tier architecture* (dreischichtige Architektur). Bei der zweischichtigen Struktur kommuniziert eine Datenschicht bzw. Datenbank direkt mit einer grafischen Benutzeroberfläche. Dabei werden die Daten ohne vorherige Bearbeitung direkt an eine Anwendung gesendet, in der diese dann verarbeitet und evtl. dargestellt werden (siehe Abb. 4.11). Diese enge Kopplung zwischen Darstellungsschicht und Datenbank führt jedoch zu Anwendungen, welche nur schwer modifizierbar sind, und die sich mit einer steigenden Zahl von Nutzern nicht einfach erweitern lässt[Edw99]. Hinzu kommt, dass Benutzeroberflächen häufig geändert werden müssen. Daher ist es ratsam, die Verarbeitungslogik vom

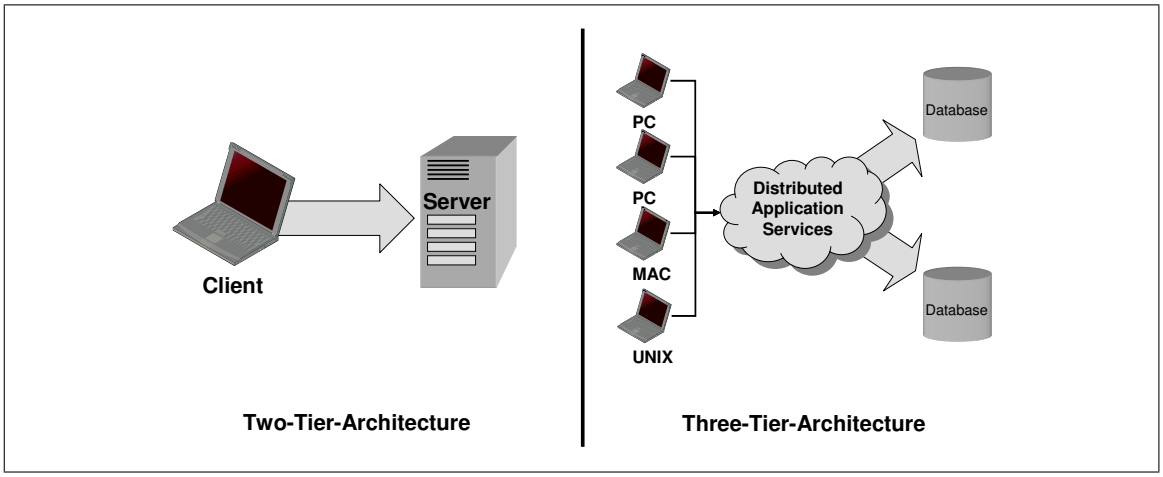


Abb. 4.11: Vergleich 2-Tier- und 3-Tier-Architektur (vgl. [Fra99])

GUI zu trennen. Insbesondere im Hinblick auf Simulationsanwendungen, bei der unterschiedlichste Modelle von den Daten profitieren sollen, ist ein Ansatz über eine zweischichtige Architektur nicht möglich.

Die Wiederverwendbarkeit von Programmcode spielt heute aus Kosten- und Zeitgründen eine große Rolle. Ein Großteil der programmierten Zeilen entfällt auf die Datenanbindungen und -verarbeitung. Diese Programminhalte sind auch bekannt unter dem Begriff der *Business Logic*. Befinden sich diese Programmabläufe alle innerhalb der oben genannten Benutzeroberflächen, so ist eine Wiederverwendung von Programmiercode nahezu ausgeschlossen. Um diese Problematik zu umgehen empfiehlt es sich die Anwendung in drei Teile aufzuspalten, anstatt in zwei (vgl. Abb. 4.11). Dieser Ansatz entspricht auch dem Model-View-Controller-Modell (MVC-Modell), welches 1979 von Trygve Reenskaug beschrieben wurde. Es handelt sich dabei um ein Architekturmuster zur Trennung von Applikationen in die drei Einheiten Datenmodell (Model) , Präsentation (View) und Programmsteuerung (Controller). Die Architektur ermöglicht ein wiederverwendbares und somit effizientes Programmdesign [Wik07f].

Vergleicht man die beiden Architekturen in Abb. 4.11, so erkennt man bei der dreischichtigen Struktur eine zusätzliche Ebene zwischen der Benutzeroberfläche und der Datenbank. Man nennt diese Ebene die *Middlewareebene*. Sie beinhaltet die gesamte Prozesslogik, die notwendig ist, um Daten von Datenbanken anzufordern, zu verarbeiten und zur Verfügung zu stellen. Der Client erhält dabei nur eine Sicht auf die Objekte der Middleware, die Datenbank erhält lediglich einen Speicher- und Abrufmechanismus für diese Objekte.

Eine dreischichtige Architektur bietet gegenüber seiner zweischichtigen Vorgänger einige wesentliche Vorteile bzgl. Geschwindigkeit, Erweiterbarkeit, Wartung und bei der Nutzung heterogener, verteilter Datenbanken [FH99]:

- Sie ist bestens geeignet für Datenbankumgebungen, bei denen die Datenbankserver und die Interaktionen mit diesen von unterschiedlicher Art und Ausprägung sind.
- Sie bietet schnelle Direktzugriffe zu mehreren Datenbanken ohne Zwischenschichten, wie sie bei der ODBC-Technik vonnöten sind.

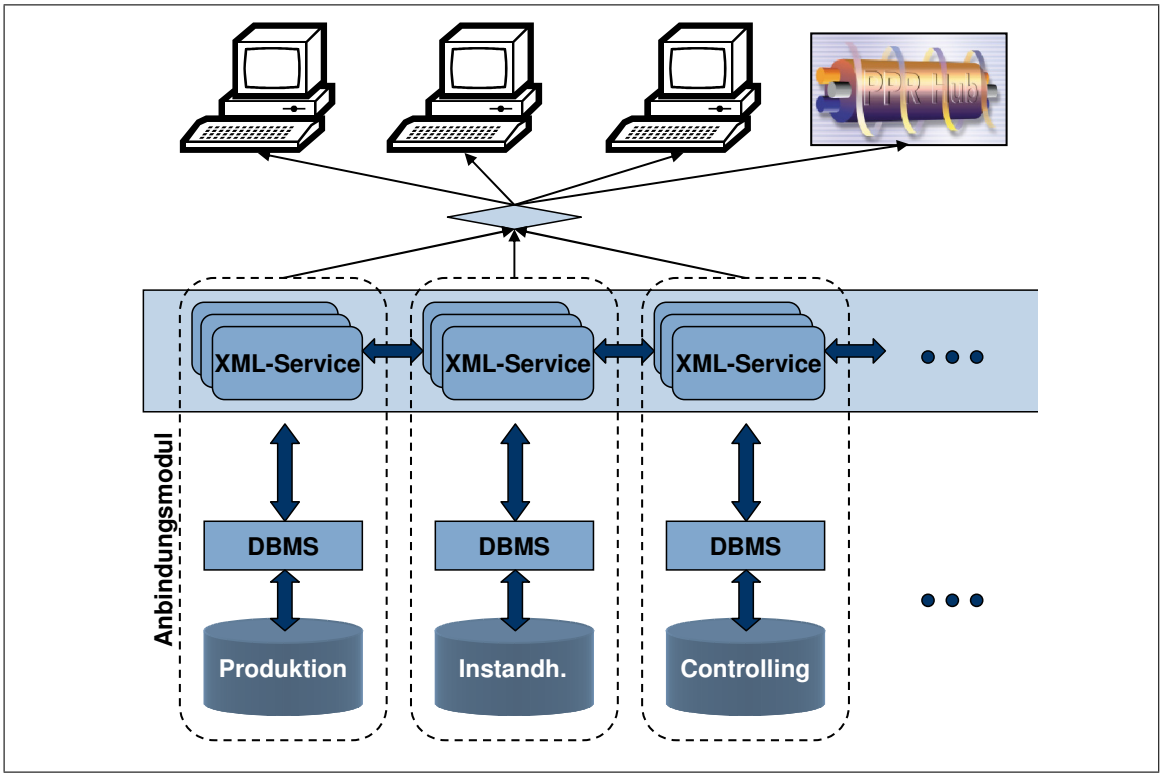


Abb. 4.12: Erweitertes Planungsdatenframework mit Datenaustausch zwischen den Modulen

- Es wird Unabhängigkeit von Datenbankhersteller und -quelle erreicht, da für alle heterogenen Zugriffe die gleiche Prozedur eingesetzt werden kann.
- Middlewaresoftware bietet dem Datenkonsumenten eine konsistente Schnittstelle zu mehreren Datenquellen (wie z.B. DBMSs, nonrelationale DBMSs und Dateisysteme).

Ein Simulationsdatenframework muss mehrere Datenbanksysteme anbinden und gleichzeitig mehrere Anwender oder Anwendungen mit Daten beliefern. Zusätzlich müssen Daten von verschiedenen Systemen logisch korrekt zusammengeführt werden. Ein schematischer Aufbau für eine solche Struktur ist in Abb. 4.12 dargestellt. In den folgenden Abschnitten soll näher auf die einzelnen Teile dieser Struktur eingegangen werden.

4.5.1 Datenbankebene

Die Datenbanken innerhalb industrieller Unternehmen bilden den Grundstein für eine effiziente Produktion. Würde eine zentrale Datenbank, wie sie der Fertigungsleitstand nutzt, ausfallen, so wäre die Produktion bei unserer heutigen Technologisierung lahmgelegt. Ein reibungsloser Ablauf muss folglich auch bei der Entwicklung der Datenbankanbindungen für das Simulationsdatenframework berücksichtigt werden. Das heißt insbesondere, dass produktionsnahe Datenbanken bei der Anforderung von umfangreichen Simulationsdaten nicht so stark mit Anfragen belastet werden, dass zeitkritische Anfragen blockiert werden. Daher soll zur Umsetzung das Connection Pooling aus Abschnitt 3.3.3 eingesetzt werden. Dabei wird eine vordefinierte

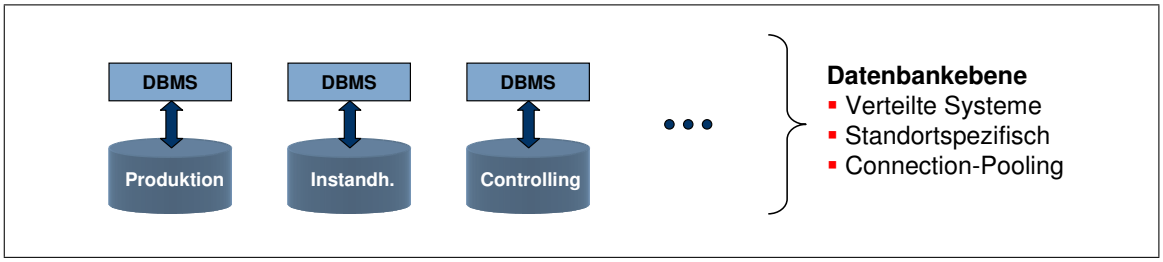


Abb. 4.13: Schematische Darstellung der Datenbankebene

Anzahl Verbindungen zur Datenbank permanent aufrechterhalten und für notwendige Anfragen verwendet. Kommen mehr Anfragen, als es Verbindungsmöglichkeiten gibt, so werden diese zeitlich nacheinander abgearbeitet. Der große Vorteil beim Connection Pooling liegt in der Zeitersparnis und dem geringeren Kommunikationsaufwand, der bei Einzelverbindungen durch den Abgleich von Parametern, wie Datenbank, Benutzer und Kennwort, entstehen würde. Diese Technik hat sich bereits in vielen Webanwendungen bewährt und wird für kommunikationsintensive Anwendungen empfohlen. In der Literatur finden sich Einsparungen in der Verbindungsanzahl zwischen 66% und 90% durch die Nutzung eines Pools [Rit98, JCKS04].

Zu den für die Simulationsanwendung interessanten Datenbanksystemen gehören:

- Dokumentationssysteme
- Produktionssysteme
- Leitstände
- Logistiksysteme
- Controllingsysteme
- CAD-Systeme

Es muss beachtet werden, dass hinter jeder Datenbank eine andere Aufbauarchitektur stehen kann. Selbst bei relationalen Tabellen kann das Datenbankdesign sehr unterschiedlich sein. Abhängig ist dies von der jeweiligen Zeit, in der die Datenbank entstanden ist, das zugehörige Datenbanksystem und natürlich den Entwickler selbst. Zwar können alle Datenbanken über die *Structured Query Language* (SQL) abgefragt werden, die Herausforderung liegt jedoch meist in den verschachtelten Tabellen und dem zugehörigen relationalen Modell.

4.5.2 Middlewareebene

Die Middlewaretechnologie wird heute in einem breiten Spektrum von Applikationen verschiedenster Art und Wiese eingesetzt. Hierzu zählen u.a. kommerzielle Transaktionsmanagementsysteme, Enterprise Application Integration (EAI) und Industriestandards zur Entwicklung von Datenbanksystemen [SB03, Deu94]. Obgleich die Middlewaretechnologie in der Softwareindustrie stetig an Bedeutung gewinnt, ist man sich in der Wissenschaft noch nicht darüber einig, was genau Middleware ist [SKS97, BWS02].

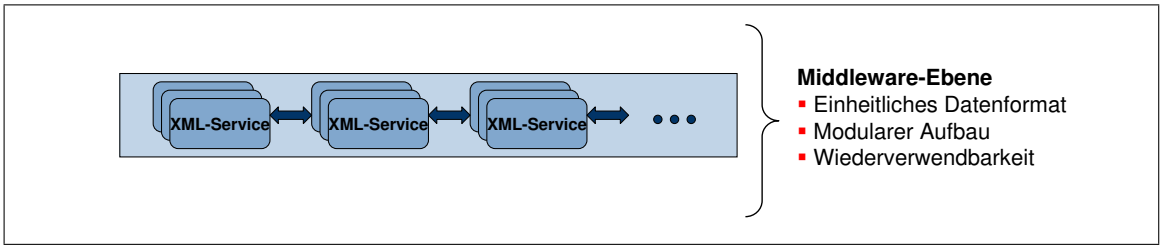


Abb. 4.14: Schematische Darstellung der Middlewareebene

Zielsetzung dieser ist stets die Anbindung an eine Datenbankanwendung. Man kann hierbei drei verschiedene Arten untergliedern [BWS02]:

- *Middle-Tier*. Der Middle-Tier läuft auf einem Server und wird oftmals als Applikationsserver bezeichnet, welcher Daten in einer dreichschichtigen Client / Server Architektur verarbeitet.
- *Verbindung zwischen Datenbanksystem und Client-Applikation*. Die Software, welche die Verbindung zwischen Anwendung und Datenbankserver herstellt, wird als Middleware bezeichnet.
- *Verbindung zwischen Datenbanksystem und Webserver*. Middleware ist die Software, welche Datenbanken mit Webservern verbindet.

Die Middleware, wie sie im Rahmen dieser Arbeit implementiert werden soll, beinhaltet die gesamte Logik, um Daten aus verschiedenen Datenbanksystemen anzufordern, zu verarbeiten und bereitzustellen. Dabei muss die Logik so gestaltet sein, dass sie stets konfigurierbar bleibt und auf diesem Wege einfache und schnelle Anpassungen erlaubt. Es muss folglich zwischen fest implementierter und variabler Logik unterschieden werden.

4.5.2.1 Statische Zusammenführungslogik

Wichtigster Bestandteil der fest implementierten Logik ist die Technik der Datenanfrage und -zusammenführung. Diese können stets nach dem selben Schema ausgeführt werden. Hierzu zählen die Anforderung einer Verbindung aus dem Connection Pool, die Übergabe der Abfrageparameter und der anschließende Empfang der angeforderten Daten. Müssen Simulationsdaten aus verschiedenen Systemen zusammengesetzt werden, so spielen die Schlüsselfelder, über welche die Daten „verbunden“ sind, eine zentrale Rolle. In dem Beispiel aus Abb. 4.1 sind zwei Tabellen in unterschiedlichen Datenbanksystemen. Der Schlüssel für die Zusammenführung der Daten ist in Tabelle A das Feld $A2$ und in Tabelle B das Feld $B1$.

Zwischen den unterschiedlichen Datenbanksystemen findet in der Regel keine Kommunikation statt. Daher müssen mehrere sequentielle Abfragen durchgeführt werden. Voraussetzung hierfür ist die Kenntnis über die Schlüsselfelder zwischen den unterschiedlichen Tabellen. Man bedient sich bei der Zusammenführung der Hashtabellen. Sie sind auch unter dem Begriff Streuwerttabellen bekannt. Diese beinhalten einen Algorithmus zur schnellen Suche von Informationen anhand eines bestimmten Schlüssels. Eine Hash Funktion konvertiert einen Eingangswert aus

einer großen Grundgesamtheit in einen Ausgangswert einer (normalerweise) kleineren Grundgesamtheit. Zu den Voraussetzungen, die jede Hash Funktion erfüllen muss, zählen [Knu73]:

- Der Eingangswert kann von beliebiger Länge sein.
- Der Ausgabewert hat eine feste Länge.
- $H(x)$ ist für jedes x einfach zu berechnen.
- $H(x)$ ist schwer umkehrbar.
- $H(x)$ ist kollisionsfrei.

Der Vorteil liegt im durchschnittlich konstanten Rechenaufwand ($O(1)$) zur Suche nach einem bestimmten Schlüssel, unabhängig davon, wie viele Einträge in der Tabelle sind. Im sehr unwahrscheinlichen, schlimmsten Fall liegt der Aufwand bei $O(n)$. Hashtabellen erzeugen für jeden Schlüssel über eine mathematische Funktion einen Hashwert, welcher auf den Ablageplatz der gespeicherten Datenstruktur im Speicher hinweist. Die Effizienz mit der in Hashtabellen abgelegte Datenstrukturen gefunden werden können und die Tatsache, dass in den verteilten Tabellen eindeutige Zuordnungsschlüssel existieren, machen diese Zusammenführungstechnik auf der Middlewareebene leistungsfähig und anwendbar.

Tabelle A	
Feld A1	Feld A2
s_1	X_1
s_2	X_1
s_3	X_2
s_4	X_2
s_5	X_2

Tabelle B	
Feld B1	Feld B2
X_1	Y_1
X_2	Y_2
X_3	Y_3
X_4	Y_4
X_5	Y_5

Ergebnis		
Feld A1	Feld A2	Feld B2
s_1	X_1	Y_1
s_2	X_1	Y_1
s_3	X_2	Y_2
s_4	X_2	Y_2
s_5	X_2	Y_2

Tab. 4.1: Schematische Darstellung der Tabellenzusammenführung

Im Folgenden soll die Vorgehensweise anhand des Beispiels aus Abb. 4.1 erläutert werden. Der Ablauflogik ist durch die Konfiguration, welche im weiteren Verlauf dieses Abschnitts noch näher beschrieben wird, bekannt, dass die Felder A2 und B1 die zusammengehörigen Schlüsselfelder bei der Zusammenführung sind. Außerdem ist es noch von Bedeutung, welche Tabelle die Haupttabelle darstellt. Das heißt, es muss festgelegt sein, welche Tabelle die Grundmenge für die Ergebnistabelle liefern soll. Es werden dann nur die Daten aus der Haupttabelle über den Schlüssel bei der Fremdtabelle abgefragt, was zur Folge haben kann, dass nicht alle Daten der Nebentabelle herangezogen werden, da evtl. Daten in dieser liegen, welche keinen Bezug zur

Haupttabelle haben. Das würde z.B. für das Datum X_5 in Spalte B1 von Tabelle B zutreffen für welches kein Schlüsselpartner in Spalte A2 von Tabelle A existiert.

In einem ersten Schritt der sequentiellen Datenzusammenführung werden nun alle Daten von Tabelle A eingelesen. Man erhält die Ergebnispaare:

$$R_A = \{\{S_1, X_1\}, \{S_2, X_1\}, \{S_3, X_2\}, \{S_4, X_2\}, \{S_5, X_2\}\} \tag{4.1}$$

Dabei wird jeder Wert aus Spalte A2 in eine Liste eingetragen, in welcher jeder Wert ohne Duplikate abgelegt wird. Nach erfolgter Abfrage hat man schließlich eine Menge:

$$M = \{X_1, X_2\} \tag{4.2}$$

Diese Ergebnismenge M übergibt man als Parameter an das DBMS, welches Tabelle B verwaltet. Aus Tabelle B erhält man die Ergebnispaare:

$$R_B = \{\{X_1, Y_1\}, \{X_2, Y_2\}\} \tag{4.3}$$

Diese werden beim satzweisen einlesen über den Schlüssel aus Spalte B1 (X_1 und X_2) in die Hashtabelle eingefügt. Angenommen man verwendet eine statische, stark vereinfachte Hash Funktion, bei der die ASCII[10]-Werte der Buchstaben aufaddiert werden:

$$h(B1) = \left(\sum_{k=1}^{Length(B1)} ascii(CharacterAt(B1, k)) \right) (mod\ 11)$$

mit

$$ascii(X) = 88,\ ascii(1) = 49\ und\ ascii(2) = 50$$

Dann erhält man die Ergebnisse in Tab. 4.2 mit deren Hashwert H.

Feld B1	Feld B2	Hashwert H
X_1	Y_1	5
X_2	Y_2	6

Tab. 4.2: Ergebnismenge aus Tabelle B mit zugehörigem Hashwert H

Jetzt könnten noch Abfragen an weitere verteilte Datenbanken über den selben oder einen anderen Schlüssel ausgeführt werden. In diesem Beispiel sollen jedoch nur zwei verschiedene

[10] American Standard Code for Information Interchange – bestimmt ganzzahlige Werte für Zeichen der Schriftsprache

Systeme schematisch dargestellt werden. Die Erweiterung auf mehrere Systeme verläuft analog.

Nach Durchführung aller definierter Abfragen wird wieder die Haupttabelle betrachtet. Sie kann nun sequentiell vom ersten bis zum letzten Satz durchlaufen werden und dabei Zeile für Zeile mit den Ergebnissen der anderen Tabellen zusammengeführt werden. Dabei wird für das Schlüsselfeld der Hashwert berechnet und mit dem passenden Wert aus Tab. 4.2 zusammengeführt. Man erhält die Ergebnisse in Tab. 4.3 und entsprechend die Ergebnismenge aus Gleichung 4.4.

Feld A1	Feld A2	$H_{A2,B1}$	Feld B1	Feld B2
S_1	X_1	5	X_1	Y_1
S_2	X_1	5	X_1	Y_1
S_3	X_2	6	X_2	Y_2
S_4	X_2	6	X_2	Y_2
S_5	X_2	6	X_2	Y_2

Tab. 4.3: Vereinigte Gesamtmenge aus Tabelle A und B über gemeinsamen Hashwert

$$R_{Ges} = \{\{S_1, X_1, Y_1\}, \{S_2, X_1, Y_1\}, \{S_3, X_2, Y_2\}, \{S_4, X_2, Y_2\}, \{S_5, X_2, Y_2\}\} \tag{4.4}$$

Ein weiterer wichtiger Punkt bei der Datenzusammenführung aus Produktionssystemen ist der physikalische Umfang der benötigten Datenmenge. Industrielle Datenbanksysteme beinhalten sehr große Datenmengen. Für die Anwendung in der Simulation spielt zwar nur ein bestimmter Ausschnitt aus diesem Spektrum eine Rolle, jedoch muss auch beachtet werden, dass die Middleware auf einem einzigen Rechner läuft, welcher nur begrenzten physikalischen Speicher zur Verfügung hat. Es ist daher wichtig schon in der Logik der Anwendung den voraussichtlichen Datenumfang zu bestimmen, bevor die einzelnen Abfragen nacheinander ausgeführt werden. Hierfür bedient man sich der sogenannten Metadaten der jeweiligen Tabellen. Datenbanktabellen bestehen aus Sätzen, welche wiederum eine feste Anzahl an Feldern beinhalten. Diese Felder bestehen, wie in der Informationsverarbeitung üblich, aus fest definierten Datentypen, wie Integer (Ganzzahl), Datum oder String (Text). Weiterhin sind für Felder variabler Länge, wie String, bestimmte Maximallängen vorgegeben. Ein Feld zum Speichern von Ortsnamen in einer Adresstabelle wird gewöhnlich auf die Länge von 50-100 Zeichen begrenzt. Die Datentypen und deren Speichergröße stellt ein Datenbanksystem in den Metadaten zur Verfügung. Eine weitere wichtige Funktion von DBMSen ist die Möglichkeit, vor dem Übertragen der Anfrageergebnisse die Anzahl der zu erwartenden Sätze zu erfragen.

In dem Beispiel aus Abb. 4.1 wird folglich in einem ersten Schritt der Speicherbedarf für das Ergebnis aus Gleichung 4.1 bestimmt. Für die Felder A1 und A2 ist der maximale Speicherbedarf über die Metadaten des Datenbanksystems bekannt. Angenommen Feld A1 beinhaltet Ganzzahlen mit den Wertebereichen 0 bis 65536, so ist der entsprechende Speicherbedarf 2 Byte. Feld 2 soll Text mit einer maximalen Länge von 50 Zeichen enthalten, entspricht also 50 Byte. Dann beträgt der maximale Speicherbedarf je Satz 52 Byte. Bevor die Daten übertragen werden, kann die Gesamtzahl der Sätze angefordert werden. In dem Beispiel aus Abb. 4.1 wären das 5 Sätze.

Man muss folglich mit einem maximalen Speicherbedarf von $M_{Tab\,A} = 5 * 52\ Byte = 260\ Byte$ rechnen.

Für die Berechnung des Speicherbedarfs der Unterabfragen müsste man die Anzahl der Schlüsselfelder aus Gleichung 4.2 kennen. Diesen Wert erhält man jedoch erst nach Durchführung der Hauptabfrage. Man kann die genannte Problematik umgehen, indem man vorab nur die Schlüsselfelder lädt, was zur Folge hat, dass man in der Regel eine wesentlich geringere Datenmenge erhält. Insbesondere bei Abfragen mit geringer Satzanzahl kann sich diese Technik jedoch negativ auf den Rechen- und Kommunikationsaufwand auswirken.

Analog zur Berechnung des Speicherbedarfs der Hauptabfrage, kann nun die maximale Datenmenge für die Unterabfragen berechnet werden. Man zieht hierfür die Anzahl der Schlüssel heran. Angenommen in Feld B1 ist der Datentyp Text mit einer Maximallänge von 50 Zeichen und in Feld B2 ist ein Datum der Länge 8 Byte gespeichert, dann beträgt der maximale Speicherbedarf $M_{Tab\,B} = 2 * 58\ Byte = 116\ Byte$. Für den gesamten Ablauf der Datenzusammenführung müsste folglich eine Speichermenge $M_Ges = 260\ Byte + 116\ Byte = 376\ Byte$ verfügbar sein.

Mit dem Verfahren zur Berechnung der notwendigen Menge an Speicher ist es möglich, bei der Überschreitung einer zu definierenden Schranke, die Abfragen aufzuteilen. Dabei genügt es jedoch nicht die Hauptabfrage in mehrere gleichgroße Teile zu zerlegen, da die Unterabfragen unter Umständen wesentlich mehr Ressourcen im Arbeitsspeicher benötigen. Es bietet sich an, die Schlüsselmenge in gleichgroße Teile zu unterteilen, da diese linear den Speicherbedarf der Unterabfragen bestimmen.

Die Zusammenführungstechnik kann auch über mehrere Unterabfrageebenen hinweg eingesetzt werden. Benötigt eine Unterabfrage, wie die auf Tabelle B in Abb. 4.1, noch Daten von einem weiteren System, so kann diese Abfrage wieder als Hauptabfrage angesehen werden, wobei letztendlich alle Abfragen rekursiv zu einem Ergebnis zusammengesetzt werden.

4.5.2.2 Variable Konfiguration

Wegen häufig wechselnder Systemumgebungen im industriellen Alltag und der sich ändernden Anforderungen an ein Datenframework durch die Simulation, ist es notwendig, die statische Ablauflogik basierend auf einer variablen Konfiguration zu gestalten. In Abschnitt 4.5.2.1 ist beschrieben, dass der Ablauf der Logik sequentiell erfolgt. Dabei sind die Möglichkeiten zur Zusammenführung in sich schachtelbar und werden rekursiv aufgelöst. Für die rechnerverwertbare Darstellung der Konfiguration bietet sich daher eine Baumstruktur an.

In Abschnitt 4.4.2 sind die Vorzüge von XML zur semantischen Darstellung von beliebigen Baumstrukturen erläutert. Auch die Ergebnisse der Datenzusammenführung werden in dieser Beschreibungssprache übermittelt. Es bietet sich daher an, auch die Konfiguration der Datenzusammenhänge in dieser Darstellungsform zu gestalten.

Zuerst sollen die zur kompletten Durchführung einer Datenzusammenführung notwendigen Merkmale aufgezeigt werden. Anhand dieser Merkmale muss es der Ablauflogik möglich sein, sowohl die zugehörigen Einzelabfragen, als auch die Zusammenführung über die Schlüsselfelder auszuführen. Im Rahmen dieser Arbeit werden physikalisch verteilte Datenbanksysteme

betrachtet. Es muss also für jede einzelne Abfrage der genaue Ort der Datenbank bekannt sein. Dieser setzt sich aus dem Rechnernamen respektive seiner Netzwerkadresse (IP) und dem Port des Datenbankdienstes auf diesem Rechner zusammen. Datenbankserver verwalten nicht selten mehrere Instanzen (z.B. eine Montage- und eine Teilefertigungsdatenbank). Daher muss auch der Name der Datenbankinstanz in der Konfiguration vorhanden sein. Mit den genannten Merkmalen ist der Ort einer Datenbank eindeutig definiert. Für die Ablauflogik ist es zusätzlich noch wichtig, die Architektur des Datenbanksystems zu kennen (z.B. IBM-DB2, Microsoft SQL-Server, etc.). Hieraus kann diese bei der Umsetzung des Konzepts die passende Implementierung auswählen. Eine exemplarische Ortsangabe, wie sie in der Konfiguration definiert sein muss, wäre exemplarisch db2://192.0.0.1:4128/Instanz1. Auf die Konfiguration wird bei der Umsetzung diese Konzepts in Kapitel 6 näher eingegangen.

Unterhalb des Ortes der Datenbank folgt die Beschreibung der Tabellen und der simulationsrelevanten Felder innerhalb dieser. In der Praxis benötigt man häufig zusammengesetzte Daten aus mehreren Tabellen einer Datenbankinstanz. Da diese Zusammenführungen jedoch lokal innerhalb des DBMS erfolgen, brauchen diese nicht in der Ablauflogik berücksichtigt werden. Sie werden in SQL formuliert und somit für die Konfiguration nur wie eine Tabelle angesehen.

Ein weiteres Merkmal, welches es zu konfigurieren gilt, sind die Schlüsselbeziehungen zwischen zwei oder mehr Datenbanken. Dabei treten Schlüssel immer paarweise auf. Man unterscheidet einen lokalen und einen Fremdschlüssel innerhalb eines entfernten Datenbanksystems. Die entsprechenden Bezeichnungen müssen identisch sein mit den Feldnamen der Tabellen. Bei verteilten Systemen muss man jedoch zwei wichtige Punkte beachten. Schlüsselfelder können in unterschiedlichen Datenbanken auch von verschiedenem Datentyp sein (z.B. Integer und String). Außerdem trifft man häufig auf das Phänomen, dass die Inhalte von Feldern in unterschiedlichen Systemen lexikalisch nicht gleich sind, obwohl sie genau dasselbe Ausdrücken (z.B. „G131“ in System A und „G_131“ in System B).

Der Problematik der verschiedenen Datentypen kann man durch Typumwandlung und falls das nicht möglich ist durch einen lexikalischen Vergleichsalgorithmus entgegenwirken. Bei unterschiedlichen Datendarstellungsarten ist dies jedoch nicht möglich. Trotzdem soll eine einfach zu konfigurierende Möglichkeit angewendet werden, welche es erlaubt Daten auf ein gleiches Format zu bringen. In einem Großteil der Fälle beschränken sich diese Ausnahmen auf vorhandene und nicht vorhandenen und nicht vorhandene Leerzeichen, manchmal trifft man aber auch auf verdrehte Datenteile. So kann es vorkommen, dass in einem System ein Feld aus einer Kombination von Baumuster und Variantencode zusammengesetzt ist, in einem anderen jedoch genau umgekehrt. Alle genannten Abweichungen kann man in der Konfiguration mit Hilfe von regulären Ausdrücken entgegenwirken. Sie stammen aus der Theorie der formalen Sprachen in der Informatik und mit ihnen können (Unter-)Mengen von Zeichenketten beschrieben werden [VW02]. In vielen Unix-Programmen werden reguläre Ausdrücke verwendet, um bestimmte Textmuster zu suchen und diese durch etwas anderes zu ersetzen. Diese formale Sprache soll bei der Datenzusammenführung zur Anpassung und Angleichung von Daten gleicher Bedeutung und anderer Darstellungsform verwendet werden.

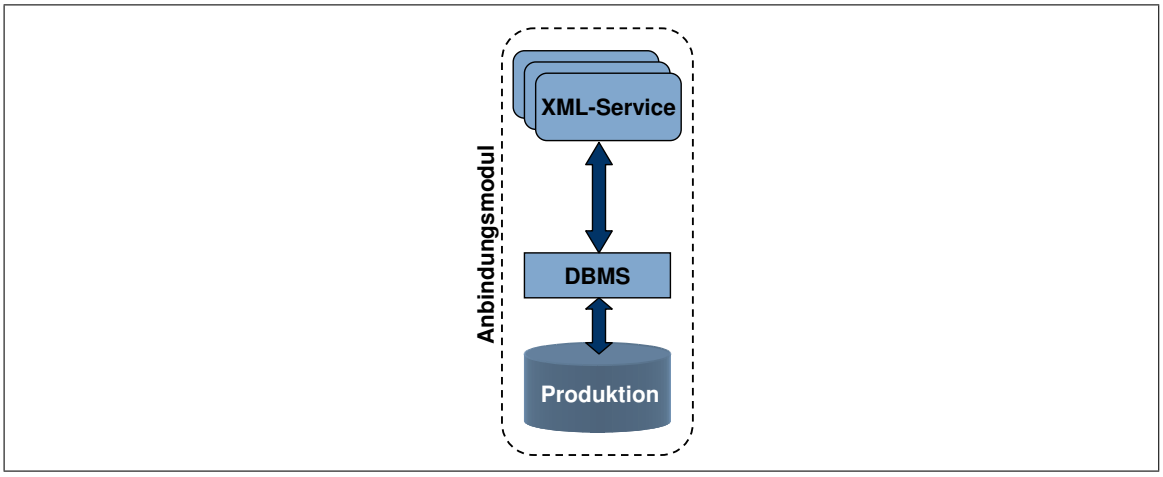


Abb. 4.15: Schematische Darstellung des Anbindungsmoduls

4.5.3 Aufbau des Anbindungsmoduls

Für den Datenaustausch zwischen dem genannten Framework und den verschiedenen DBMSen wird ein allgemeingültiges Anbindungsmodul benötigt (vgl. Abb. 4.15). Die in der Konfiguration der Middlewareebene angegebene Datenbankinstanz muss innerhalb dieses Moduls bzgl. der Kommunikation umgesetzt werden. Dabei spielt, besonders bei einer Vielzahl von Zugriffen, wie sie durch die Datenzusammenführung entstehen können, die Leistungsfähigkeit eine besondere Rolle. Stand der Technik bei der effizienten Anbindung von Datenbanken sind die in Abschnitt 3.3.3 beschriebenen Connection-Pools. Beim Starten der Webanwendung wird eine bestimmte, vorkonfigurierte Anzahl von unabhängigen Verbindungen zu einer Datenbank hergestellt. Diese werden innerhalb eines Pools verwaltet und bei Bedarf zur Verfügung gestellt. Der Effizienzgewinn gegenüber proprietären Anbindungstechniken liegt dabei in der ersparten Zeit für den Verbindungsauf- und Verbindungsabbau. Sie verbrauchen unnötig Zeit, da Netzwerkverbindungen hergestellt, Benutzerdaten ausgetauscht und Kennwörter abgeglichen werden müssen.

In der Konfiguration des Datenframeworks sind alle benötigten Datenbankinstanzen mit den jeweiligen Verbindungs- und Benutzerdaten hinterlegt. Es müssen lediglich noch Parameter ergänzt werden, welche die Verwaltung des Verbindungspools benötigt. Hierzu zählen unter anderem die minimale und maximale Anzahl offener Verbindungen, die maximale Leerlaufzeit von Verbindungen und die längste Wartezeit auf eine Verbindung, im Fall dass alle Kommunikationskanäle bereits belegt sind. Diese Angaben tragen im Wesentlichen zur Leistungsfähigkeit der Datenanbindung bei und müssen daher in der variablen Konfiguration ergänzt werden.

Das Anbindungsmodul setzt die Anfragen der Middlewareebene in für die Datenbanksysteme ausführbare SQL-Befehle um, und übergibt diese an das DMBS. Dieses führt den SQL-Befehl aus und liefert dem Anbindungsmodul eine Datenmenge zurück. Das Resultat wird dabei weiterhin im Anbindungsmodul verwaltet, welches vorwärts und rückwärts durch die Ergebnisse springen kann. Das Module für die Datenzusammenführung ruft diese Daten bei Bedarf ab und setzt sie anschließend um. Nach erfolgter Datenübergabe kann vom Anbindungsmodul die Ver-

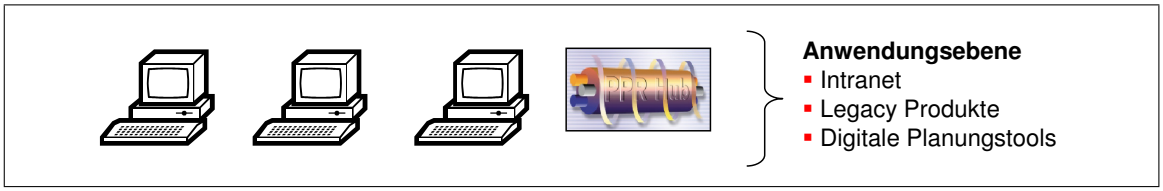


Abb. 4.16: Schematische Darstellung der Anwendungsebene

bindung wieder zurück in das Pool gegeben werden. Die Verbindung steht dann für weitere Anfragen zur Verfügung.

4.5.4 Anwendungsebene

Auf der obersten Ebene des in Abb. 4.12 dargestellten Frameworks befinden sich die Datenkonsumenten. Im Vordergrund steht hierbei die Datenversorgung von Simulationssystemen und der teilautomatisierte Simulationsmodellaufbau. Nachdem eine konsistente und einheitliche Datensicht zu systemübergreifenden Daten geschaffen ist, müssen diese auf der Anwendungsebene in ein von einem Simulator lesbares Format transferiert werden.

Das Konzept soll jedoch nicht nur Simulationssysteme mit Informationen versorgen, sondern auch die Möglichkeit bieten, diverse Informationssysteme und weitere Anwendungen mit Planungsdaten zu beliefern. Die diversifizierte Datenmenge und der logische Zusammenhang simulationsrelevanter Daten stellen ein umfangreiches Abbild der Daten zur Produktionsplanung dar. Diese werden in den Unternehmen oftmals in unterschiedlichen Systemen gehalten und somit nicht sinnvoll zusammengeführt und dargestellt. Im Bereich der Simulationsanwendung in Produktion und Logistik bedarf es jedoch ebendieser Datenumfänge, um ein realitätstreues und aussagekräftiges Modell zu generieren.

In der Produktions- und Logistikplanung liegt das Bestreben der Automobilindustrie und auch anderer Industriezweige in der Schaffung und dem Einsatz einer einheitlichen Planungsdatenstruktur, welche die Methodik des Simultaneous Engineering unterstützt. Softwarepakete, wie der Process Engineer von Delmia und der EM-Planner von Tecnomatix, sollen diese Methodik unterstützen und somit eine effiziente und schnelle Planung erleichtern. Auch diese Planungswerkzeuge müssen durch Schnittstellen zur bestehenden Systemwelt versorgt werden, um einen geregelten Datenaustausch zwischen Planungs- und Produktions-/Dokumentationssystemen zu bieten. Daher soll auch die Integration dieser Anwendungen mit Hilfe des Datenframeworks berücksichtigt werden. Auf diesen Aspekt soll bei den Praxisbeispielen in Kapitel 7 nochmals näher eingegangen werden.

4.5.5 Konfiguration der Datenzusammenhänge

Bisher wurde die Semantik der Datenbereitstellung nach außen, also zu anderen Systemen, wie der automatisierten Modellgenerierung für die Simulation, beschrieben. Die Einheitlichkeit und

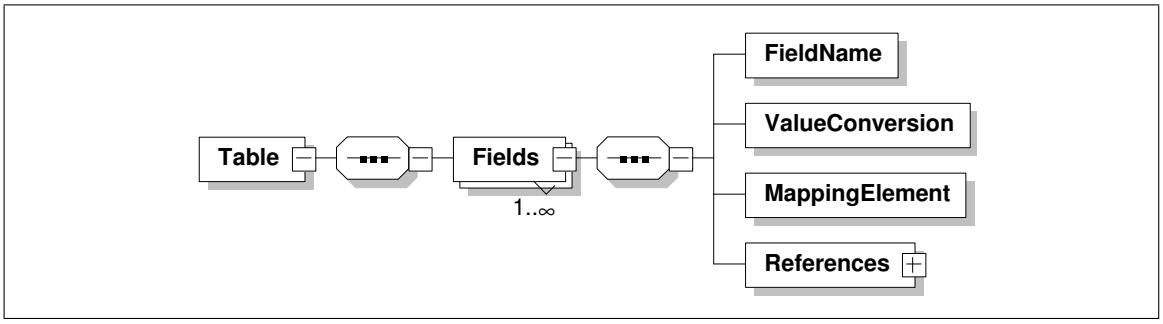


Abb. 4.17: Konfiguration des Mapping-Elements

die stets gleiche Darstellung der Informationen in XML machen eine Weiterverarbeitung in anderen Anwendungen trivial. Der aufwändigere Teil bei der Entwicklung eines Simulationsdatenframeworks entfällt jedoch auf die Methodik der Datensammlung und -verarbeitung. Nachdem festgelegt ist, wie die Inhalte dargestellt werden, muss nun eine Systematik definiert werden, wie man die Zusammenhänge zwischen den verschiedenen Datenbanksystemen und deren Inhalte beschreiben kann, so dass sie automatisiert und flexibel von den Datenkonsumenten verwendet werden können.

Nach außen sehen die Daten durch die XML-Darstellung stets gleich aus. Wenn ein Unternehmen beschließen sollte, eines Tages statt eines ERP Systems der Firma SAP zukünftig ein anderes System einzusetzen, so bleibt letztlich eine Jahresstückzahl für ein Teil in Bezug auf die Aussagekraft das gleiche. Systemseitig kann diese jedoch in völlig anderer Form implementiert sein. Für die Datenabnehmer spielt dies keine Rolle, da in XML das Element weiterhin `Jahresstückzahl` heißt und eine Zahl beinhaltet. Bei einem Systemwechsel oder einem Update muss lediglich die Konfiguration aus Abschnitt 4.5.2.2 verändert werden. Schlimmstenfalls muss zusätzlich ein neues Anbindungsmodul aus Abschnitt 4.5.3 für das neue System bereitgestellt werden.

Jetzt kann es jedoch vorkommen, dass besagte Jahresstückzahl im alten System unter dem Feldnamen `JStueck` gespeichert war und im neuen unter dem Feldnamen `PartsPerYear` abgelegt wird. Der SQL-Befehl kann in der variablen Konfiguration angepasst werden, aber die Middleware weiß nicht mehr, welchem Element sie diese Daten in XML zuordnen soll. In Abschnitt 4.4 ist die Notwendigkeit für ein standardisiertes Austauschformat und ein Lösungsansatz hierfür beschrieben. Die Daten sollen nach außen immer gleich aussehen und lediglich um weitere Elemente ergänzt werden können. Es muss jedoch noch eine Konfiguration der Zuordnung zwischen den Daten auf Datenbankebene und denen der standardisierten Form definiert werden.

Hierfür bedient man sich eines Zuordnungselements innerhalb der variablen Konfiguration. Hiermit wird den Feldern innerhalb einer Abfrage, die für das Simulationsdatenframework benötigt werden, ein entsprechendes Element in der XML-Struktur zugeordnet. Das Zuordnungselement, auch Mapping-Element genannt, ist in Abb. 4.17 dargestellt. Am Beispiel der Jahresstückzahl würde im Mapping-Element der Inhalt *Modell* → *Teile* → *Jahresstückzahl* stehen. Die Konfiguration der systemübergreifenden Datenzusammenhänge wird bei der Umsetzung des Konzepts in Kapitel 6 noch näher beschrieben.

Kapitel 5

Abstraktion von Produktionsabläufen für die Simulation

Im Verlauf dieser Arbeit wird in Kapitel 2 die Notwendigkeit zur Integration der Materialflusssimulation in den Planungsablauf der Industrie aufgezeigt. In Kapitel 3 werden die informationstechnischen Zusammenhänge und Möglichkeiten zur Zusammenführung dieser beschrieben. Ein Verfahren zur Aufbereitung von simulationsrelevanten Daten aus unterschiedlichen Datenbanksystemen wird in Kapitel 4 hergeleitet und beschrieben. Dieses schafft die Grundlage zur Versorgung von Simulationsmodellen mit den notwendigen Daten aus den Planungs- und Produktionssystemen. In den folgenden Abschnitten werden die notwendigen Vorbereitungen und Anpassungen innerhalb der Simulationssysteme erläutert, welche dann die Grundlage schaffen um in Kapitel 6 Modelle teilautomatisiert zu Generieren. Hierzu werden erst die wichtigsten Grundbausteine, aus welchen ein Modell erzeugt werden kann, definiert und anschließend die übergeordneten logischen Abläufe hergeleitet, so dass diese in einem Simulationssystem implementiert werden können.

5.1 Bausteinbibliothek für Simulationsentitäten

Es reicht noch nicht aus, die simulationsrelevanten Daten einer Fabrik in einer effizienten Struktur zur Verfügung zu stellen. Die vorhandenen Daten müssen in für die Simulation verwertbare Entitäten überführt werden. Diese stellen Objekte innerhalb eines Modells dar. Man unterscheidet permanente (z.B. Maschinen) und temporäre (z.B. Teile) Entitäten und sie können eine Untermenge anderer Entitäten enthalten. Die objektorientierte Simulation ist am leistungsstärksten, wenn der Anwender bei der Modellierung alle Komponenten als unabhängige Entitätsklassen abbildet [Kil03]. Um eine möglichst realitätstreue Modellierung zu erreichen, werden verschiedene flexible Basiselemente für den Simulator definiert. Die Basisentitäten können in zwei grundlegende Arten unterteilt werden [RJ03].

5.1.1 Entität Maschine

Maschinen können im Allgemeinen gut abstrahiert werden, da man sie als „Black Box“ betrachten kann. Das heißt, ein Teil muss auf eine bestimmte Art und Weise in die Maschine kommen, wird dort unabhängig von der realen Fertigungstechnologie eine bestimmte Zeitdauer festgehalten und muss anschließend wieder aus der Maschine entfernt werden. Diese Abstraktion ist natürlich nur gültig, wenn man automatisierte Fertigungsmaschinen betrachtet, die z.B. über

eine NC-Steuerung verfügen und unabhängig von einer äußeren Beeinflussung Teile fertigen können. Prinzipiell kann man drei verschiedene Be- und Entladestrategien unterscheiden:

1. Handbeladen:
   Der Werker muss nach jedem Bearbeitungsprozess ein neues Teil in die Maschine einlegen oder einspannen. Solange der Werker nach dem Bearbeitungsprozess mit anderen Tätigkeiten beschäftigt ist, ensteht bei der Ressource Maschine eine Brachzeit.

2. Bandlader:
   Die Maschine wird über ein Transportband mit Teilen versorgt. Es obliegt dem Werker, wann und wie viele Teile er nachlegt. Die Ressource besitzt die Fähigkeit, Schichtpausen zu überbrücken (siehe Abb. 5.1).

3. Automatisierter Belader:
   Der Werker kann eine ganze innerbetriebliche Transporteinheit (ITE) an der Maschine bereitstellen. Die Ressource bedient sich mittels Greiferarme selbst neuer Teile und stellt die Halbfertigprodukte nach Bearbeitung in einer anderen ITE wieder bereit.

Die unterschiedlichen Laderkonzepte unterscheiden sich insbesondere im Bedarf an menschlicher Arbeitszeit. Die Menschzeiten sind in den innerbetrieblichen DV-Systemen in REFA-Zeitbausteinen [REF91] erfasst und können für jeden einzelnen Prozess abgerufen werden. Anhand der Inventarnummer können online die Prozesse abgerufen werden, welche vom Produktionsverfahren, den benötigten Spannmitteln und Werkzeugen auf einer Maschine produziert werden können, oder welche von der Planung auf einer bestimmten Maschine zur Produktion vorgesehen sind.

Einen weiteren und teilweise auch großen Zeitbaustein bilden die Rüstzeiten. Diese sind in der Regel nicht konstant und hängen von verschiedenen Faktoren, wie z.B. der Verfügbarkeit und dem Zustand der Werkzeuge ab. Daher werden die Rüstzeiten nicht als feste Zeitbestandteile in den DV-Systemen dokumentiert. Um diese Daten automatisiert beschaffen zu können, muss man den Umweg über die dokumentierten Echtzeiten der Vergangenheit gehen und daraus mittels statistischer Verfahren eine Verteilung bilden.

Ein Prüfarbeitsplatz ist in der Regel direkt an eine Maschine gebunden oder wird in Ausnahmefällen von einer Maschinengruppe gemeinsam genutzt. Für die Abstraktion auf Modellebene kann ein solcher Arbeitsplatz wie eine handbeladene Maschine modelliert werden. Je nach Prüfverfahren kann dabei die werkerunabhängige Maschinenzeit $t = 0$ gesetzt werden, sofern der Werker das Prüfen selbst ausführt oder die Messtechnik überwachen muss.

5.1.2 Entität Werker

Durch den hohen Automatisierungsgrad in der Getriebefertigung bei der DaimlerChrysler AG werden Werker vorwiegend im Rahmen von Gruppenarbeit oder einer Mehrmaschinenbedienung eingesetzt. Dabei versorgt ein Werker eine bestimmte Maschine mit Teilen und während der unabhängigen Laufzeit dieser, kann er bereits die nächste Maschine bestücken. So ist es je

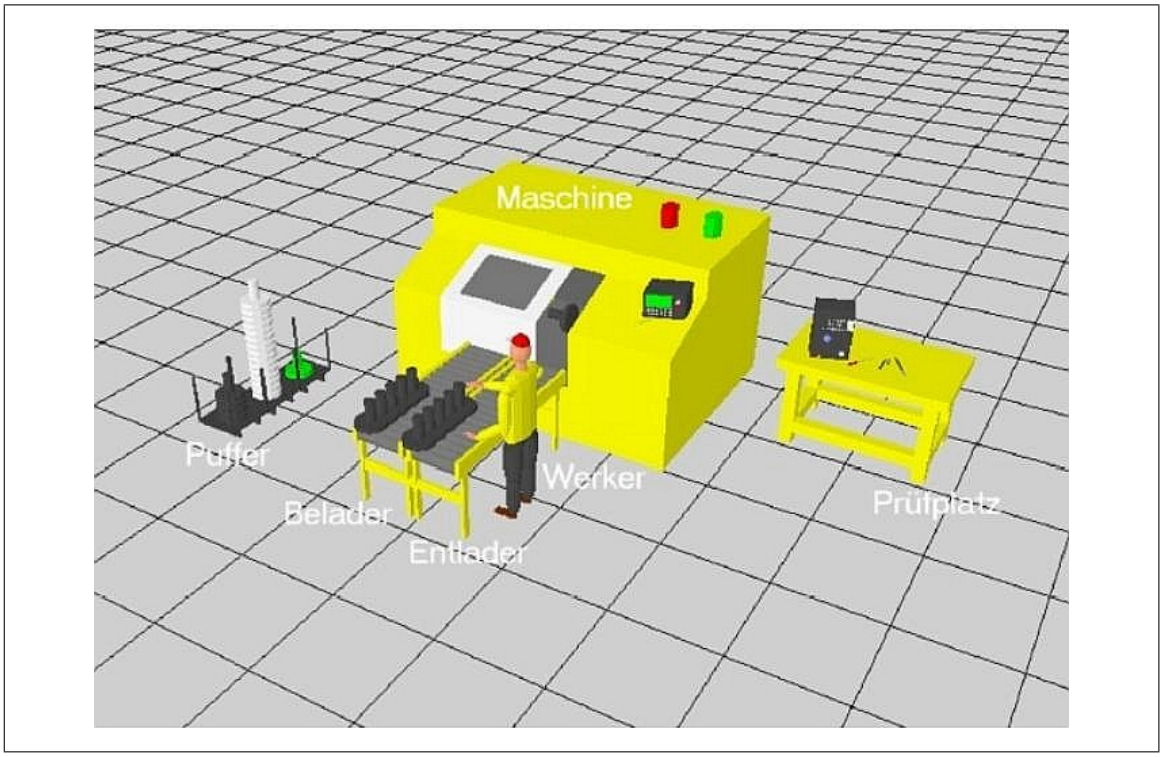


Abb. 5.1: Bandbeladene Maschine mit zugehörigem Prüfplatz [RJ03]

nach Automatisierungsgrad und unabhängiger Laufzeiten möglich, dass bis zu sieben Maschinen von einem Werker bedient werden können. Man spricht hier auch von einem Mehrmaschinenfaktor (MS). Bei sieben bedienbaren Maschinen wäre folglich $MS = 7$.

Betrachtet man die Entität Werker im Rahmen der teilautomatisierten Modellgenerierung, so zeigt sich schnell, dass in keinem DV-System definiert ist, welcher bestimmte Werker eine vorgegebene Maschine bedienen kann. Vielmehr obliegt es dem Meister in der Produktion, seine Arbeitskräfte effizient und nach Qualifikation einzusetzen. Für die Umsetzung des beschriebenen Konzepts stellt dies mit das größte Hindernis dar. Im Simulationsmodell muss ein Pool aus Werkern modelliert werden, aus welchem dann unter Berücksichtigung von Pausen- und persönlichen Verteilzeiten einzelne Simulationsentitäten den Prozessen zugewiesen werden. Die Qualifikationen und persönlichen Einsatzmöglichkeiten, wie sie in der Realität vorhanden sind, können dabei nicht automatisiert berücksichtigt werden. Um auch diese sinnvoll abbilden zu können muss eine manuelle Zuordnung anhand einer Zuweisungsmatrix getroffen werden (vgl. Tab. 5.1).

	Werker 1	Werker 2	Werker 3
Prozess 1	0	0	9
Prozess 2	9	5	0
Prozess 3	0	1	0

Tab. 5.1: Zuweisungsmatrix Werker zu Prozessen mit Prioritäten 0–9

Das Beispiel der Werkerzuordnung ist nur eines im Bereich der Diskrepanz zwischen in Systemen gehaltenen Daten über Fertigungsabläufe und der realen Produktion. Es zeigt, dass bei heutigem Detaillierungsgrad der Planungsdaten ein vollautomatischer Ansatz zur Modellgenerierung nicht möglich ist. Daher wurde in dieser Arbeit der Schwerpunkt auf die teilautomatische Generierung von Simulationsmodellen gelegt. Dem Simulationsexperten soll hierdurch ein Teil seiner manuellen Tätigkeiten abgenommen werden und dem Produktionsplaner mit Grundkenntnissen in der Materialflusssimulation der Einstieg in die Nutzung dieser bei seinen täglichen Aufgaben ermöglicht werden.

5.1.3 Logische Verbindungen zwischen Simulationsentitäten

Der technologisch bedingte Durchlauf eines einzelnen Teils durch die Fertigung ist in den jeweiligen DV-Systemen anhand einer aufsteigenden Arbeitsvorgangsnummer (Avo-Nr.) erkennbar. Jeder Arbeitsvorgang ist wiederum verknüpft mit einer Teilenummer und einer Inventarnummer. Über diese Verknüpfung ist es möglich, die gesamten Verbindungen innerhalb eines Modells zu ermitteln und somit die gerichteten Graphen für das Teilerouting zu bestimmen. XML eignet sich aufgrund seiner Aufbaustruktur mit Kanten und Knoten sehr gut zur Darstellung von Graphen und Netzen, wie sie hier beschrieben sind. Es hat sich bereits gezeigt, dass es sehr nützlich sein kann, einen Prozessablauf, wie in einer Kostenstelle, grafisch mit den zugehörigen Prozesszeiten darzustellen, um einen Gesamtüberblick von einem bestimmten Bereich zu erhalten. Das Framework für die automatisierte Modellgenerierung kann somit auch als Planungs- und Management-Informationssystem genutzt werden.

Viele Teile können technisch bedingt bei einem bestimmten Bearbeitungsschritt auf unterschiedlichen Maschinen gefertigt werden. Der Verbindungsgraph ist zwar gerichtet, jedoch nicht eindeutig festgelegt. Das erfordert wiederum eine entsprechende Logik zur Teileweitergabe von einem Prozess zum nächsten. Hierbei sind unterschiedliche Strategien denkbar. Es kann die nächste freie, die bereits am längsten stillstehende, die schon gerüstete Maschine, etc. ausgewählt werden. Man muss hier die reale Strategie beobachten, um festzustellen, wie ein Meister in der Produktion diese Aufgabe durch Erfahrungswerte einschätzt und steuert. Die meisten Maschinen sind jedoch in hohem Maße automatisiert und erfordern somit auch höhere Rüstzeiten. Daher genügt es in einem ersten Schritt, die rüstoptimale Logik unter Berücksichtigung der nächsten freien Maschine zu implementieren. Unterschiedliche Strategien können in XML-Attributen hinterlegt werden. Beim späteren Umwandeln der XML-Dateien in Simulationsmodelle müssen die entsprechenden Attribute wieder in ein für den Simulator ausführbares Format überführt werden.

5.1.4 Steuerungslogik des Simulationsmodells

Aufbauend auf die in Abschnitt 5.1.3 erläuterten Verbindungen zwischen Simulationsentitäten muss nun die übergeordnete Steuerung des Simulationsmodells entwickelt werden. Bisher sind die einzelnen Bausteine aus der zugehörigen Bibliothek bzgl. deren Verhalten in einem Simulationsmodell erstellt und die Teileweitergabe zwischen diesen geregelt worden. Um einen Ablauf eines automatisch generierten Simulationsmodells zu gewährleisten, müssen noch bestimmte

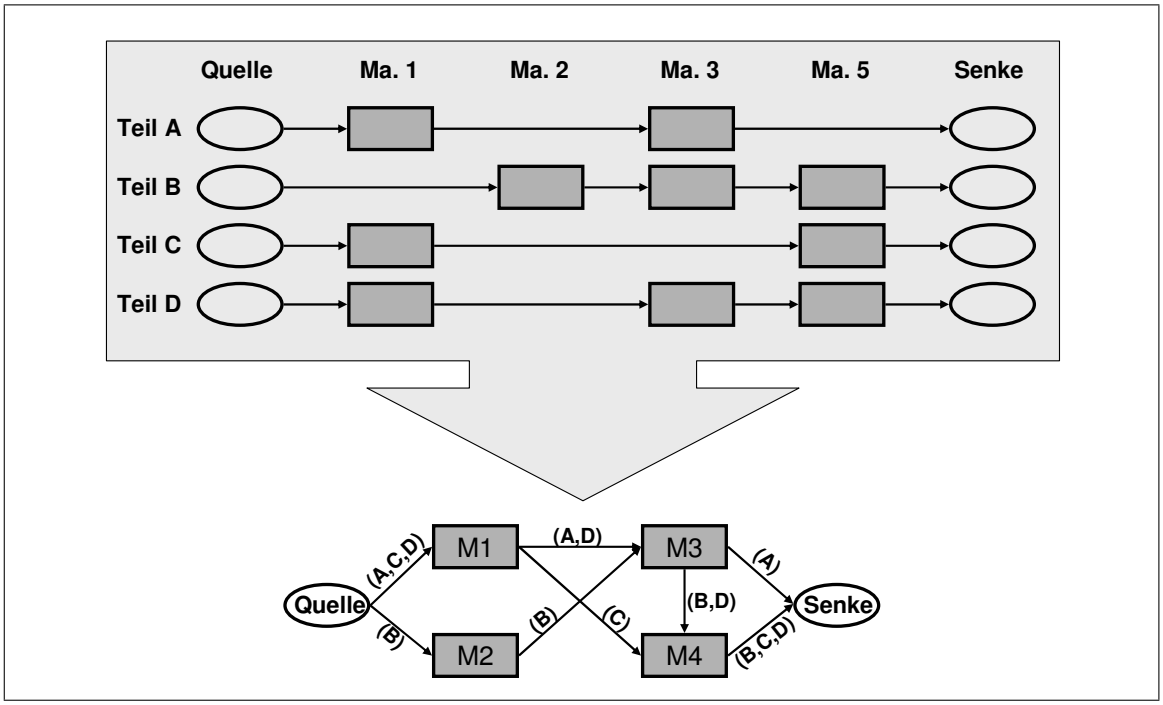


Abb. 5.2: Überlagerung einzelner Graphen zur Bestimmung des Verbindungsnetzwerks

übergeordnete Rahmenbedingungen erfüllt sein. Dabei soll als erstes der Materialfluss betrachtet werden.

Es gibt in der realen Fertigung verschiedene Arten der Materialsteuerung. Hierzu zählen die klassischen Steuerungsarten Push und Pull [SZ92]. Eine Push-Steuerung stellt das einfachste Prinzip der Produktionssteuerung dar. Idealisiert kann man sagen, dass Rohmaterial in den Puffer vor der ersten Bearbeitungsstation kommt, anschließend auf dieser bearbeitet wird und danach in den Puffer vor der nächsten Maschine eingeht (vgl. Abb. 5.3). Bei dem Pull-Prinzip wird die Produktion durch eintreffende Aufträge ausgelöst. Die Entnahme der einzelnen Vorprodukte löst dann die Produktion in den vorgelagerten Fertigungsstufen aus. Teile werden also nur an Arbeitsstationen geliefert, wenn sie dort gebraucht werden [Zij00]. Dieses Prinzip wird auch „Produktion auf Abruf“ genannt. Im Idealfall werden dadurch keine Zwischenlagerbestände aufgebaut und die Lieferbereitschaft wird aufrechterhalten. Man spricht daher auch von bedarfssynchroner Fertigung [DD02].

Eine mögliche Ausprägung des Pull-Prinzips ist Kanban. Die Funktionsweise dieses Steuerungsmechanismus wird anhand des Ablaufs bei der Produktion von Teilen, die mehrere Maschinen in einer fest vorgegebenen Folge zu durchlaufen haben, kurz dargestellt. Vor jeder Arbeitsstation existiert ein Puffer mit wenigen Teilen jedes Produkts, das auf der entsprechenden Arbeitsstation bearbeitet werden soll. An jedem Teil oder Ladungsträger ist eine Kanban-Karte befestigt. Wenn an einer Maschine ein Bearbeitungsvorgang für ein bestimmtes Produkt durchgeführt werden soll, werden die benötigten Teile oder Komponenten dem Puffer entnommen. Die zugehörigen Kanban-Karten werden an die vorhergehende Arbeitsstation weitergereicht, um dort die Bearbeitung der entsprechenden Teile auszulösen und den Eingangspuffer der an-

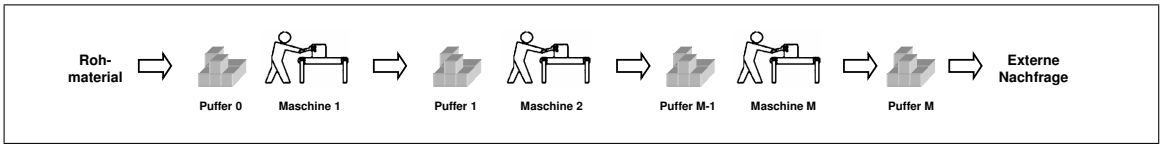


Abb. 5.3: Ablauf in einer Push-Linie [MHDE05]

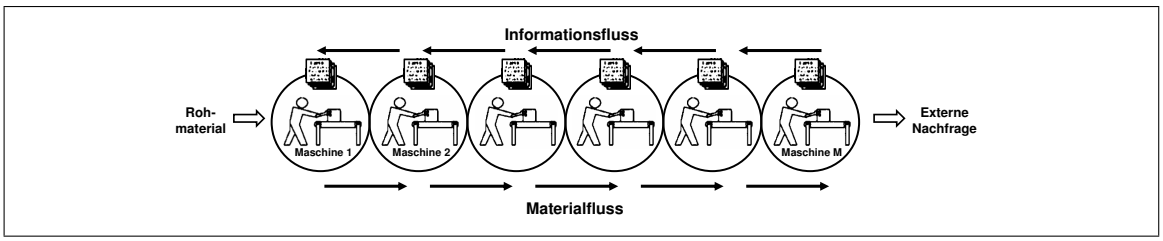


Abb. 5.4: Schematische Darstellung einer Kanbansteuerung [MHDE05]

fordernden Maschine wieder aufzufüllen. Dazu werden beim Vorgänger die erforderlichen Teile wiederum dem Eingangspuffer entnommen und die Kanban-Karte an dessen Vorgänger gesandt (vgl. Abb. 5.4). Eine Endnachfrage löst so eine Folge von „Auffüllbestellungen“ entlang der gesamten Fertigungskette bis hin zur Materialbeschaffung aus [Zij00].

Damit ein solches System überhaupt funktioniert, sind einige Voraussetzungen zu erfüllen, die eventuell erst durch technische und organisatorische Änderungen geschaffen werden müssen [DD02]. So sollte die Nachfrage keinen großen Schwankungen unterworfen sein. Außerdem ist bei der Produktion ein hoher Wiederholungsgrad erforderlich. Deshalb ist Kanban besonders bei Sorten- und Serienfertigung mit Fließablauf geeignet. Bei der Einführung von Kanban ist somit eventuell eine Umstrukturierung der Produktionsprozesse notwendig, um eine Reihen- oder „Quasi-Fließfertigung“ zu erreichen.

Ferner sind kurze Rüstzeiten und damit niedrige Rüstkosten nötig, da eine Bestandsverringerung bei Kanban über kleine Losgrößen und kurze Durchlaufzeiten erzielt wird. Speziell diese Prämissen machen die Einführung des Steuerungssystems in einer Getriebeproduktion meist schwierig. Einerseits sind die Rüstzeiten in der Dreh- und Verzahnungsbearbeitung sehr hoch (1-5 Stunden) und andererseits gibt es sehr viele Varianten, welche eine Serienfertigung stören. Letztlich sollten die Anlagen hohe Zuverlässigkeit und Stabilität aufweisen, was durch Qualitätssicherungsmaßnahmen während der Produktion und eine sorgfältige Instandhaltung erreicht werden kann.

Der Erfolg von Kanban liegt in der Einfachheit des Systems. Es wird kein zentrales PPS-System benötigt, welches die Produktion steuert, denn diese erfolgt dezentral durch vermaschte Regelkreise. Der Hauptnutzen liegt in der Reduzierung der Durchlaufzeiten und Bestände im Vergleich zu einer nach dem Pushprinzip gesteuerten Fertigung. Man muss jedoch beachten, dass diese Faktoren nur mit erhöhtem Rüstaufwand erreicht werden können. Somit muss der Schnittpunkt der beiden gegenläufigen Funktionen aus Bestands- und Rüstkosten gefunden werden.

Ein PPS-System ist die zentrale Komponente bei der Produktionssteuerung nach dem Push-Prinzip. Im Gegensatz zum Pull-Verfahren wird ein Fertigungsprozess nicht durch einen Kun-

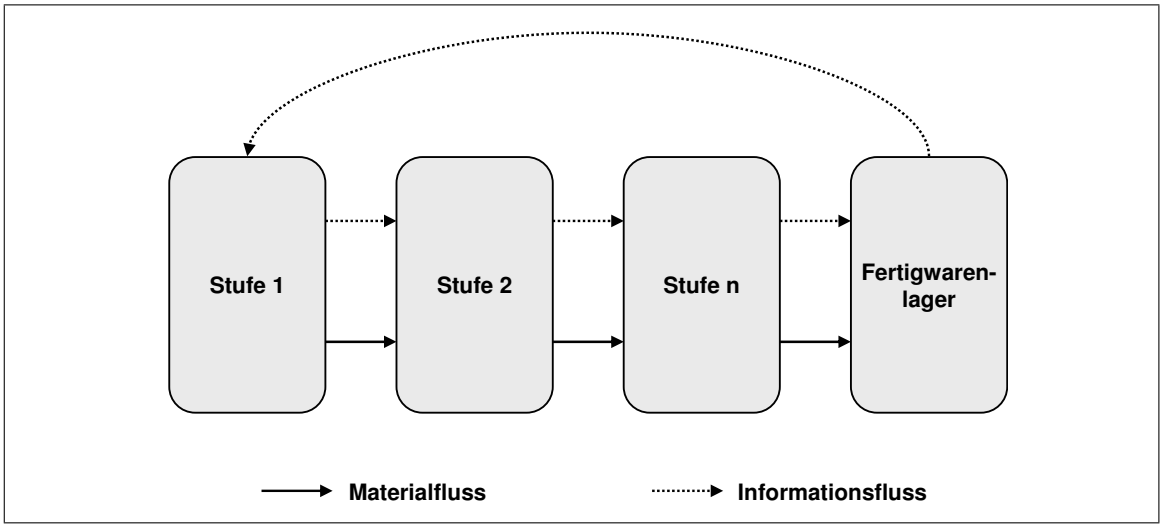


Abb. 5.5: Schematische Darstellung der ConWIP-Steuerung [GT00]

denauftrag angestoßen. Die Bedarfe für die zu planenden Perioden werden mittels Prognosemethoden aus Daten der Vergangenheit und Marktanalysen gewonnen. Die Bedarfe an Endprodukten werden als Planprimärbedarfe bezeichnet. Danach führt das System eine Stücklistenauflösung der Endprodukte durch und generiert sogenannten Sekundärbedarf. Unter dem Sekundärbedarf versteht man die Menge der Rohteile und Komponenten, die zur Fertigung der Endprodukte benötigt werden. Handelt es sich bei einer Komponente um ein Zukaufteil, wird von dem PPS-System eine Bestellanforderung für Einkaufsmaterial erzeugt. Bei eigengefertigten Teilen erfolgt die Generierung eines Fertigungsauftrags. Aufgrund der voreingestellten, erfahrungsbasierten Durchlaufzeit und der aktuellen Kapazitätsauslastung wird der Starttermin für den Fertigungsauftrag bestimmt und die gesamtkostenminimale Losgröße berechnet. Bei Erreichen des Starttermins wird die Produktion angestoßen. Nach der Produktion wird das Endprodukt solange auf Lager gelegt bis ein Kundenauftrag eintrifft.

Die Schwierigkeit bei der Fertigung nach dem Push-Prinzip liegt in der Komplexität der Steuerung. Es wird ein System benötigt, welches die Planung und Terminierung der Fertigungsaufträge durchführt. Andererseits führen Fehler in der Absatzprognose zu hohen Lagerbeständen oder auch zu Lieferausfällen. Außerdem erweist sich die Steuerung nach dem Push-Prinzip als sehr unflexibel, da auf eine kurzfristige Änderung des Produktionsprogramms (z.B. Eilauftrag) nur unzureichend reagiert werden kann.

Um die Vorteile beider Fertigungsprinzipien zu nutzen, bietet sich die Fertigungssteuerung nach der ConWIP-Methode an. Sie ist eine Mischform aus Push- und Pull-Fertigung. Während im Kanban-System für jede Fertigungsstation und jede Erzeugnisart ein separater Regelkreis eingerichtet werden muss, wird bei ConWIP für die gesamte Produktionslinie nur ein Kartenkreislauf eingerichtet (vgl. Abb. 5.5). Es warten also Gebinde mit einer festgelegten Menge an Material in einem Fertigproduktlager, dem sogenannten Supermarkt, auf Aufträge. Die Entnahme eines Endprodukts löst einen neuen Produktionsauftrag der erstbearbeitenden Station aus. Der Rohmaterialnachschub in das System erfolgt also nach dem Pull-Prinzip. Innerhalb der Produktionslinie wird nach dem Push-Prinzip vorgegangen [GT00].

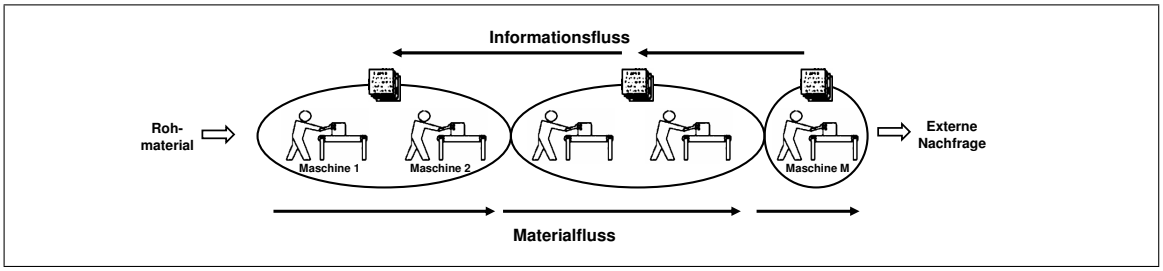


Abb. 5.6: Schematische Darstellung einer segmentierten ConWIP-Steuerung [MHDE05]

Eine Zwischenform von ConWIP und Kanban ist die segmentierte ConWIP-Steuerung (vgl. Abb. 5.6). Die Information wird wie bei Kanban von Bearbeitungsstation zu Bearbeitungsstation weitergegeben. Allerdings sind nicht die einzelnen Bearbeitungsstationen durch Supermärkte voneinander getrennt, sondern ganze Produktionslinien. Innerhalb einer Linie wird das Material nach dem Push-Prinzip weitergereicht [MHDE05]. Diese Kombination der Push- und Pull-Steuerung ermöglicht gleichzeitig eine flexible und zeitnahe Einlastung von Produktionsaufträgen, eine Verringerung von Beständen, aber auch eine kostengünstige Losgrößenfertigung. Für den Werker in der Fabrik ist das System einfach zu handhaben, da er feste Losgrößen an die nächstbearbeitende Maschine weitergibt und dabei ohne Karten, wie sie beim Kanban üblich sind, auskommt. Lediglich an Bereichsgrenzen, wie z.B. an der erstbearbeitenden Maschine, muss anhand der Kanban-Karten das nächste zu fertigende Los bestimmt werden. Dieser Vorteil mag zwar im ersten Moment verschwindend gering erscheinen, doch hat es sich in der Teilefertigung, in der mit Öl, Emulsion, Waschmaschinen, etc. gearbeitet wird, gezeigt, dass Karten in Maschinen geraten, verschmutzt werden oder auf sonstige Weise verschwinden und somit das gesamte System gestört wird.

Ein weiterer Punkt bei der Implementierung einer Steuerungslogik ist das Generieren von Aufträgen. In der realen Fertigung fungiert die Montage als Kunde der Teilefertigung. Es gibt verschiedene Absatzprognosen für die jeweiligen Produkte und Varianten, die je nach Zeitraum unterschiedliche Detaillierungsgrade aufweisen. In der Automobilindustrie werden Fertigungsplanungen meist auf Basis einer Absatzprognose für die nächsten 3-5 Jahre durchgeführt. Die Produktion selbst stützt sich eher auf eine Prognose für die nächsten 6 Monate bis 2 Jahre. Je kürzer der Prognosezeitraum wird, desto detaillierter und genauer wird auch die Aufschlüsselung der Varianten und Baumuster. Die Montage selbst arbeitet mit einer starren Zeitleiste von 2-10 Tagen, je nach Produkt, Durchlaufzeit und Kunde.

Für die Simulationsanwendung ist das feste Produktionsprogramm der nächsten Stunden oder Tage interessant, wenn es darum geht verschiedene Produktionsalternativen zu vergleichen oder verschiedene Schichtmodelle zu prüfen. Im Bereich der Planung von Fertigungsanlagen ist jedoch meist die Prognose für die nächsten 6-12 Monate von Bedeutung. Grund hierfür ist die Tatsache, dass bei einer Fertigung mit großen Losen eine längere Zeitschiene simuliert werden muss, damit auch bestimmte „Exoten“, welche nur selten produziert werden müssen, in der statistischen Auswertung berücksichtigt werden. Diese Prämisse macht es jedoch schwierig, in der Simulation ein realitätsnahes Senkenverhalten nachzubilden, da über einen längeren Zeitraum keine realistischen Annahmen getroffen werden können, auf welche Art und Weise die Montage Teile verbrauchen wird. Daher empfiehlt es sich in einem ersten Schritt lediglich ei-

ne Gleichverteilung anzunehmen. Das kann durch einen getakteten Prozess erfolgen, welcher die Zwischenankunftszeit (engl. Interarrivaltime (IAT)) darstellt. Sie berechnet sich nach Formel 5.1. Man kann dabei jedes Teil einzeln betrachten, oder auch Teilefamilien gruppieren, um bestimmte Baumuster oder Varianten abzubilden. Der Simulationsexperte kann im weiteren Verlauf der Planungsphase bestimmte Erfahrungswerte oder geplante Kundenbelieferungen in Form von anderen Verteilungen im Modell implementieren.

$$\begin{aligned} IAT &= \frac{\sum_{i=1}^{n} p_i}{s} \\ s &= \text{Jahresarbeitszeit} \\ n &= \text{Teilefamilien} \\ p_i &= \text{Jahresstückzahl Teilefamilie i} \end{aligned} \tag{5.1}$$

Die Steuerungslogik in Form eines segmentierten ConWIP und das Abzugsverhalten von Teilen und Losen sind somit definiert und können implementiert werden. Damit sind die Rahmenbedingungen für die Implementierung einer teilautomatisierten Modellgenerierung geschaffen. Im folgenden Kapitel sollen die beschriebenen Grundlagen umgesetzt und im Simulationsbetrieb getestet werden.

5.2 Simulationsschwerpunkte in der Teilefertigung

In Abschnitt 2.3 auf Seite 17 sind bereits einige Schwerpunkte der Simulationsanwendung in der teilefertigenden Industrie genannt worden. Simulationsanwendungen zur Erreichung einer Rüstzeitenminimierung, einer Losgrößenoptimierung und einer bestmöglichen Verteilung von Werkern auf Maschinengruppen stehen dabei im Vordergrund.

Im Bereich der Teilefertigung kann davon ausgegangen werden, dass in den meisten Fällen eine Losgrößenfertigung und ein technologisch bestimmter Vorranggraph den Ablauf in der realen Fertigung bestimmt. Zu den wichtigsten Ressourcen gehören Werker, Fertigungs- und Prüfmaschinen, automatisierte Be- und Entladetechnik und Transporthilfsmittel. Anhand dieser Grundvoraussetzungen soll im weiteren Verlauf dieser Arbeit eine Abstrahierung dieser Fertigungsabläufe erfolgen und beschrieben werden. Sie soll genutzt werden, um eine teilautomatisierte Modellgenerierung zu ermöglichen, welche basierend auf den Daten des Simulationsdatenframeworks aus Kapitel 4 Simulationsläufe ermöglicht.

5.2.1 Schematische Abstraktion der Teilefertigung

Bestimmte Bereiche der realen Teilefertigung können in der Simulation so gestaltet werden, dass Modelle nach einem wiederverwendbaren Schema aufgebaut werden, welches mit unterschiedlichen Daten, wie Rüst-, Bearbeitungs- und Beladezeiten hinterlegt wird. In Abb. 5.7 ist dargestellt, wie ein gesamtes Produktionssystem über mehrere Stufen in seine einzelnen Objekte zerlegt werden kann. Diese Abstraktion wird von Simulationsexperten durchgeführt, um Modelle der Realität zu erhalten und in ein rechnerausführbares Modell zu überführen. In der Teilefertigung kann man die technischen Rahmenbedingungen nutzen, um ein Modell in Form einer Matrixstruktur analog zu Abb. 5.7 zu generieren. Jedes Teil hat einen spezifischen, technologisch bedingten Ablauf durch die Produktion (z.B. Drehen → Fräsen → Waschen → etc.). Gleichzeitig existieren für die verschiedenen Fertigungsverfahren jeweils eine oder mehrere Ressourcen, auf welchen dieser Schritt ausgeführt werden kann. Man ordnet nun die Ressourcen der jeweiligen Fertigungsschritte auf den Spalten einer Matrix an und definiert die technischen Durchlaufmöglichkeiten der Teile auf den Zeilen der Matrix. Bei mehreren verschiedenen Teilen erhält man folglich eine mehrdimensionale Matrix. Diese beinhaltet den gerichteten Graphen aus Abb. 5.2 auf Seite 70.

In dem exemplarischen Simulationsmodell aus Abb. 5.8 spiegelt sich auch die ConWIP-Steuerung, welche in Abb. 5.5 auf Seite 73 dargestellt ist, wider. Es sind zwei Fertigungsbereiche dargestellt, durch welche Material nach dem Push- und Losgrößen-FIFO (First In First Out)-Prinzip durchgeschleust wird, und zwischen den Fertigungsbereichen befindet sich ein Supermarkt als Puffer. Losgrößen-FIFO bedeutet, dass nicht jedes einzelne Teil sondern eine ganze Losgröße innerhalb der Warteschlange betrachtet wird. Ebenso befindet sich nach der Fertigung und vor der Montage ein Puffer zur Versorgung von Montagebedarfen und zum Ausgleich einer losgrößengesteuerten Fertigung.

Ein Ausschnitt der simulationsrelevanten Daten ist beispielhaft in Abb. 5.9 dargestellt. In dieser Matrix sind alle Bearbeitungszeiten für die unterliegenden Simulationsentitäten hinterlegt.

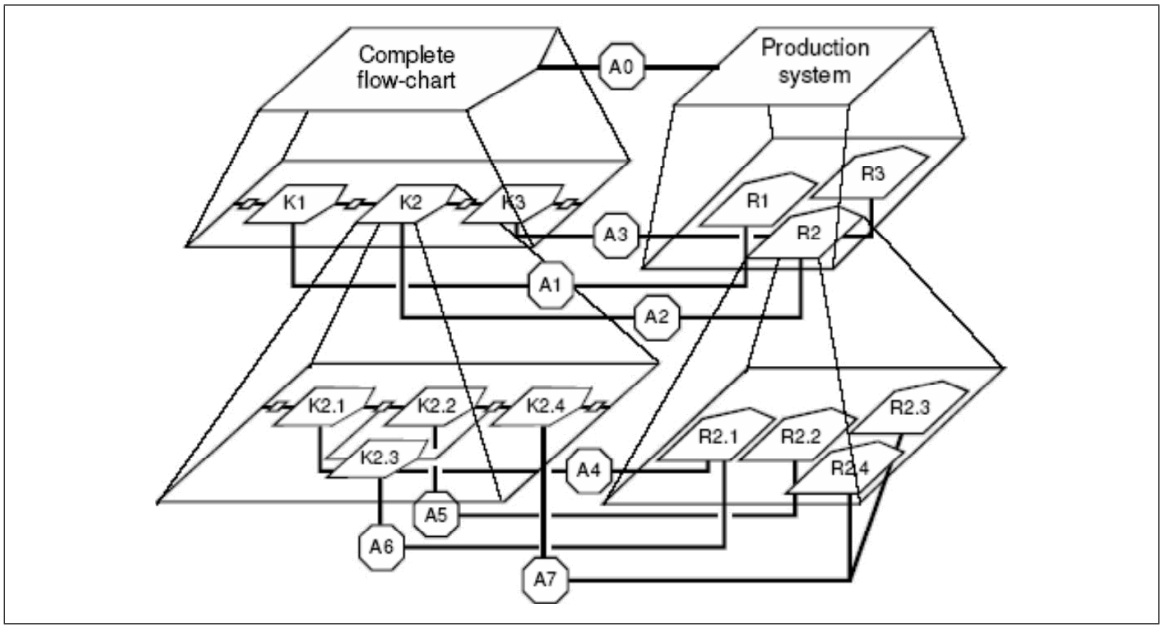


Abb. 5.7: Hierarchisches Modell eines Produktionssystems [ZFJ00]

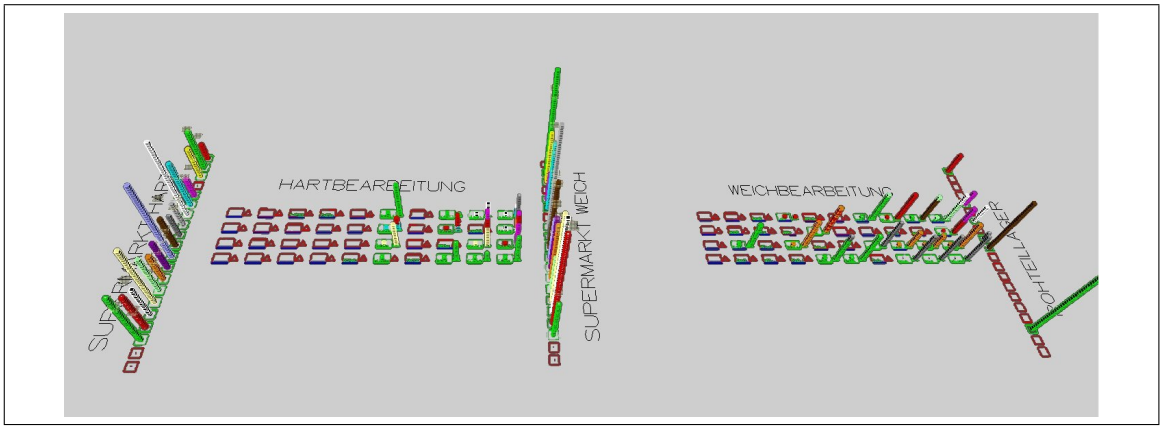


Abb. 5.8: Beispiel für den Matrixaufbau eines Modells in Automod

Bearbeitungszeiten pro Maschine und Teil [min] (WEICH)

Masch / Typ	A1632620002	A1632620005	A1632620105	A1632630001	A2032620702	A2032622505
1	0,5	1,15	1,15	0,9	0,5	1,08
2	0,5	1,15	1,15	0,9	0,5	1,08
3	9999	2,17	2,17	1,7	9999	2,7
4	9999	2,17	2,17	1,7	9999	2,7
5	9999	1,1	2,1	1,7	9999	1
6	1,92	9999	9999	9999	1,79	9999
7	9999	9999	9999	9999	9999	1,4
8	0,85	9999	9999	9999	0,85	9999
9	3,5	9999	9999	9999	2,75	9999
10	3,5	9999	9999	9999	2,75	9999
11	9999	9999	9999	9999	9999	4,3
12	9999	9999	9999	9999	9999	4,3
13	9999	2,17	2,1	1,7	9999	2,7
14	9999	2,1	2,1	2,1	9999	2,05
15	9999	0,9	0,9	0,54	9999	0,9
16	9999	0,9	0,9	0,54	9999	0,9
17	9999	1,88	1,88	1,85	9999	1,88
18	9999	1,88	1,88	1,85	9999	1,88
19	9999	1	1	1	9999	1
20	9999	9999	9999	9999	9999	9999

Abb. 5.9: Beispiel für Bearbeitungszeiten innerhalb der Matrixstruktur

Diese Datenmatrizen werden ebenso wie für die Bearbeitungszeiten auch für Rüst-, Ausfall und Reparaturdauern, sowie für Jahresstückzahlen, Losgrößen und weitere Daten benötigt.

Das Element „Bearbeitungsstation“ welches bisher ein Element der beschriebenen Matrix ausmacht, kann und muss noch weiter abstrahiert werden, um den Anforderungen der realen Fabrik möglichst genau zu entsprechen. Eine Bearbeitungsstationen innerhalb einer Teilefertigung lässt sich nicht durch ein einzelnes Simulationsobjekt darstellen, da an ihr mehrere Arbeitsschritte ausgeführt werden. Sie können in folgende Einzelschritte zerlegt werden:

1. Eine Untermenge (z.B. Bodenroller) einer Losgröße wird in die Warteschlange vor der Maschine gestellt.
2. Die Warteschlange wird im Losgrößen-FIFO-Prinzip abgearbeitet (gleiche Teile haben Priorität vor anderen wegen teilweise hohen Rüstzeiten).
3. Rüsten der Maschine mit Werkzeugen, Spannmitteln, NC-Steuerung, etc.
4. Das Transportmittel wird manuell der Beladetechnik zugeführt.
5. Start der Bearbeitung.
6. Berücksichtigung von Werkzeugwechseln, Störungen und Ausfällen (statistisch oder diskret).
7. Nachladen von Teilen (optional).
8. Prüfen anhand vorgegebener Prüfschärfe[1] (Laufzeitparallel oder Laufzeitunterbrechung).
9. Abtransport aus Entladetechnik (bei Bedarf).
10. Wiederholung ab 6. bis Los abgearbeitet ist.
11. Start nächstes Los bei 3.

Für das Simulationsmodell bedeutet dies, dass die Bearbeitungsstation aus mehreren Simulationsentitäten zusammengesetzt werden muss. Zu den wichtigsten gehören der Puffer vor der Maschine, je nach Art der Beladetechnik ein Band, Robotik oder bei Handbeladung kein Belader, die Maschine selbst, Entladetechnik analog der Beladetechnik, ein Prüfplatz und ein Puffer nach der Maschine. Der Puffer nach der Maschine ist innerhalb der Arbeitsabfolge identisch mit dem Puffer vor der nächsten Maschine. Je nach Aufbau und Logik des eingesetzten Simulators kann es noch andere relevanten Entitäten geben, z.B. für statistische Auswertungen oder die Einbindung menschlicher Interaktionen, welche jedoch für die Abstraktion nicht von Bedeutung sind. Die beschriebene Basisgruppe aus Einzelentitäten ist schematisch in Abb. 5.10 dargestellt. Dieses Grundelement stellt die Basis für den automatisierten Aufbau eines Produktionsverbunds dar.

Bei der Umsetzung in dem Simulationssystem Quest, wie es in Abb. 5.1 dargestellt ist, wurden in einem ersten Schritt die Basisentitäten in einem leeren Modell erzeugt. Sie entsprechen der schematischen Darstellung in Abb. 5.10. Man kann in der genannten Abbildung den Eingangspuffer erkennen, welcher eine eigene Klasse bildet. Abhängig von der Anzahl der Teile, welche später

[1] Verhältnis Anzahl ungeprüfte zu geprüfte Teile

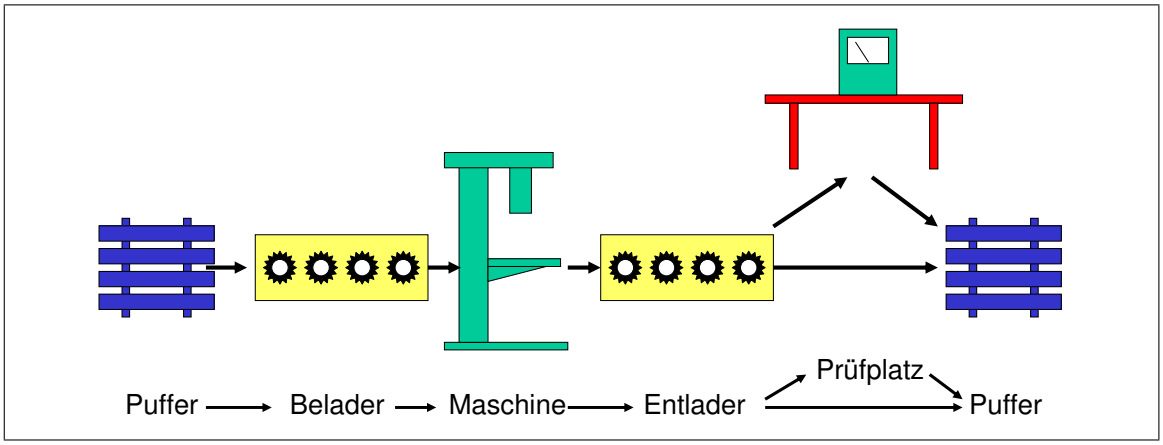


Abb. 5.10: Darstellung einer Bearbeitungsstation aus Einzelentitäten

auf der dargestellten Maschine gefertigt werden sollen, wird dieser Klasse bei der Modellgenerierung die Anzahl der zugehörigen Elemente zugeordnet. Die Bibliotheksbausteine sind nur ein Grundgerüst, welches dann zusammen mit den Daten aus dem Datenframework dem Anwendungsfall angepasst werden. In Abb. 5.1 könnten folglich drei unterschiedliche Teile produziert werden. Die Be- und Entlader werden in Quest als Transportbänder (Klasse Conveyor) realisiert und bei der Initialisierung des Modells mit den Kapazitäten aus den Realdaten versorgt. In dem genannten Beispiel wäre die Laderkapazität fünf Teile. Anschließend folgt die Klasse Maschine zur Abbildung der Bearbeitungsstation. Auch der Prüfplatz ist in Quest als Maschine modelliert, da dieser sowohl eine bestimmte Zeit benötigt und die Ressource Werker benötigt.

Zur Umsetzung der in diesem Abschnitt genannten, logischen Einzelschritte im Ablauf muss in Quest die Simulation Control Language (SCL) benutzt werden. In dieser Programmiersprache wird die Logik des Simulationsmodells programmiert. Jeder der beschriebenen Einzelschritte wird innerhalb des Programmcodes umgesetzt, so dass der Ablauf des Submodells zwischen Ein- und Ausgang sichergestellt ist und im weiteren Verlauf der Arbeit als „Black Box“ verwendet werden kann.

Dieses in sich abgeschlossene Modell wird als solches abgelegt und kann später mehrfach in ein übergeordnetes Modell geladen werden. In sich ist diese noch nicht lauffähig, da es keine Quelle, Senke und keinen Werker besitzt. Diese Entitäten können nicht in dem abgeschlossenen Submodell vorhanden sein, da diese nicht in jedem benötigt werden, sondern erst im Gesamtmodell von Bedeutung sind. In den folgenden Abschnitten wird beschrieben, wie aus diesen Submodellen ein Gesamtmodell erstellt werden kann.

5.2.2 Zusammenführung zu einem Produktionsverbund

In der bereits definierten Matrixstruktur innerhalb eines zu generierenden Simulationsmodells ist dargestellt, dass einerseits Teile auf verschiedenen Ressourcen gefertigt und andererseits, bedingt durch den technologischen Ablauf, über mehrere verschiedene Ressourcenverbünde durch

die Produktion geschleust werden können. Es ist folglich möglich, die in Abschnitt 5.2.1 definierte Basisgruppe zu einem Verbund zusammenzuführen. Dabei können zwei Stufen unterschieden werden:

1. Zusammenführung auf Teileebene
2. Zusammenführung auf Technologieebene

Die Austauschbarkeit eines Basiselements auf Teileebene bedeutet, dass ein spezielles Teil auf mehreren Ressourcen unter gleichen, oder ähnlichen Bedingungen gefertigt werden kann. Gleiche Bedingungen wären dann der Fall, wenn mehrere exakt baugleiche Maschine in einer Fertigung vorhanden sind, welche auch gleiche Betriebsmittel, Werkzeuge und NC-Programme verwenden. In diesem Fall handelt es sich quasi um ein geklontes Basiselement. Unter den Begriff der Ähnlichkeit würden unterschiedliche Werkzeuge, ungleiche Bearbeitungszeiten oder andere Beladungsmodalitäten fallen. Diese Unterscheidung ist insbesondere im Hinblick auf die Zielsetzung einer Simulationsstudie wichtig. Es kann in der Planung durchaus vorkommen, dass man höhere Bestände in Kauf nimmt, um die Summe der Rüstzeiten zu reduzieren, da Bestände kaufmännisch nur zu einem bestimmten Prozentanteil der Gesamtkosten kalkuliert werden. Die Kosten für Rüsten spiegeln sich jedoch direkt in Löhnen wider und beeinflussen das operative Ergebnis direkt.

Die Zusammenführung auf Technologieebene geht einen Schritt weiter und fasst die Elemente der Teilebenen bei gleicher Technologie zusammen. Innerhalb einer dieser Stufen befinden sich alle Maschinen für einen Prozess (z.B. Drehen). Angenommen das Drehen ist der erste Prozess im Fertigungsablauf, so definiert die Matrixstruktur innerhalb der ersten Spalte die Zuordnungsmöglichkeit aller in der Simulation vorhandenen Teile eines Produktionsbereichs zu den entsprechenden Maschinen. Folglich spiegelt sich in dieser Zuordnung auch der in Abschnitt 5.1.3 definierte logische Durchlauf von Teilen durch eine Teilefertigung wider.

In einem weiteren Schritt werden mehrere Technologieverbünde zu einem gesamten Produktionsnetzwerk zusammengeführt. Hierzu werden, ungeachtet der Tatsache, dass nicht alle Teile die selben Bearbeitungsschritte durchlaufen, alle möglichen Verbünde hintereinander aufgereiht. Man erhält ein mehrstufiges Modell, wie es exemplarisch in Abb. 5.11 dargestellt ist.

5.2.3 Logischer Ablauf

In Abschnitt 5.2.2 wird durch Verkettung mehrerer Basiselemente aus Abschnitt 5.2.1 ein Produktionsverbund für die abstrakte Modellierung erzeugt. Unter Bezugnahme auf die Steuerungslogik aus Abschnitt 5.1.4 stellt dieser den Bereich der Push-Steuerung innerhalb eines segmentierten ConWIP-Konzepts dar. Dieser entspricht in der realen Fertigung einer Kostenstelle oder Meisterei.

Zum besseren Verständnis kann die logische Steuerung auch grafisch mithilfe von Petrinetzen dargestellt werden. Diese dienen im Operations Research der Abbildung logischer Zusammenhänge und gehören zu der Methode der Netzplantechnik [NM02]. Ein Netzplan ist gemäß DIN 69 900 definiert als „die grafische Darstellung von Ablaufstrukturen, die logische und zeitliche

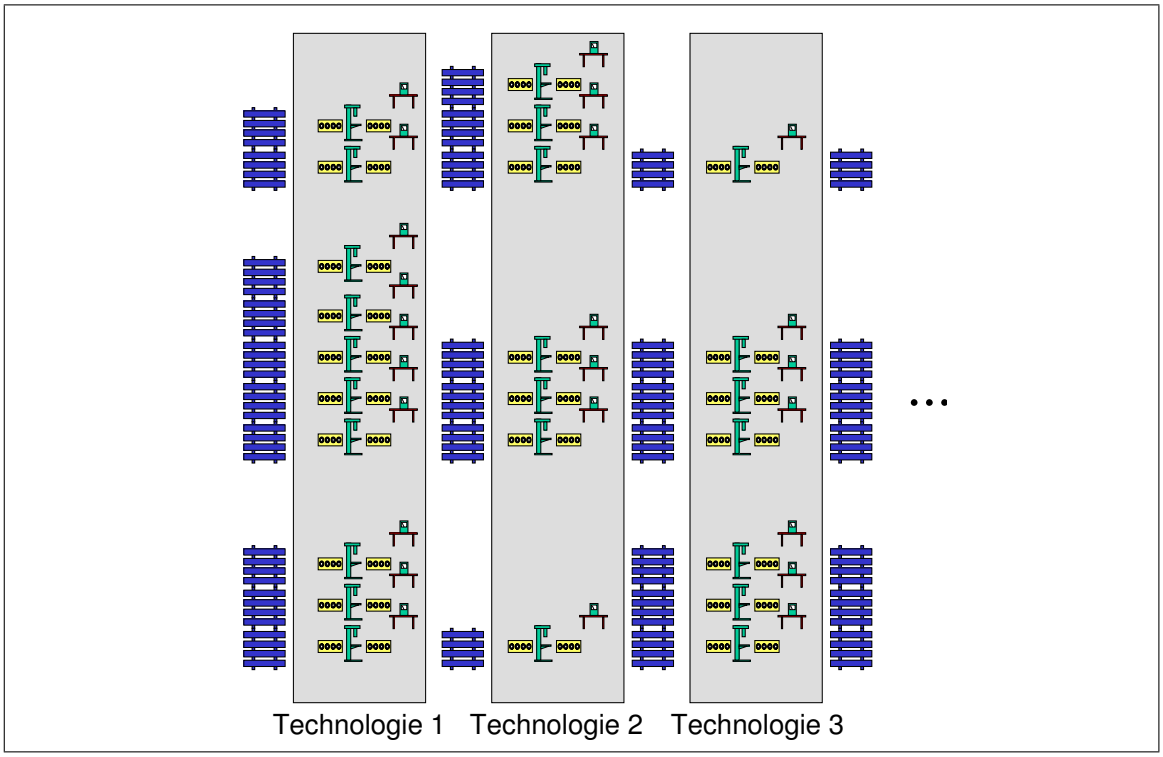


Abb. 5.11: Exemplarische Darstellung eines Verbunds aus Basisgruppen

Aufeinanderfolge von Vorgängen veranschaulicht“ [DIN87] (vgl. auch Abb. 5.12). Durch die Abstraktion eines Simulationsmodells, wie es in den vorangegangenen Abschnitten erläutert ist, erreicht man auch eine abstrahierte logische Steuerung. Der in Abb. 5.2 dargestellte Verbindungsgraph zeigt zwar das entsprechende Teilerouting durch die Fertigung, jedoch noch nicht den vollständigen logischen Ablauf. Diesen kann man anhand eines Petrinetzes darstellen, weil dieses Abhängigkeiten und Vorränge in der Fertigung berücksichtigen kann. So können Abhängigkeiten, wie dass ein Werker verfügbar sein muss, um Teile zur nächsten Maschine zu bringen, oder dass eine leere Transporteinheit vorhanden sein muss, bevor Teile produziert werden können, abgebildet werden.

Die Fertigungsschritte müssen zuerst für jedes mögliche Teil in ein Knoten- und Kanten-Modell übertragen werden, welches bei Überlagerung fertigbarer Teile das Netzwerk aller möglichen Abläufe bildet. Bearbeitende Stationen (Maschinen) bilden die Knoten, das Teilerouting bildet die Kanten. Zusätzlich können bestimmte Ablaufanforderungen, wie ein Werker oder ein Betriebsmittel, in das Netz aufgenommen werden. In Abb. 5.12 ist ein exemplarisches Petrinetz eines synchronisierten Prozesses zweier Maschinen mit einem gemeinsam genutzten Betriebsmittel dargestellt. In der Praxis stellt es dann für den Simulationsexperten keine Schwierigkeit mehr dar, diesen Netzplan in einem gängigen Simulationssystem, wie Quest, Automod oder Simflex/3D in ein ausführbares Simulationsmodell zu überführen. [RVJ03]

Eine Kombination aus Push-Steuerung und FIFO-Konzept ermöglichen es, in der Ablauflogik eines entsprechenden Simulators eine allgemeingültige Funktion zu implementieren, welche für jede beliebige Fertigungsstufe die Teileweitergabe regelt. Sobald eine beliebige Ressource zur

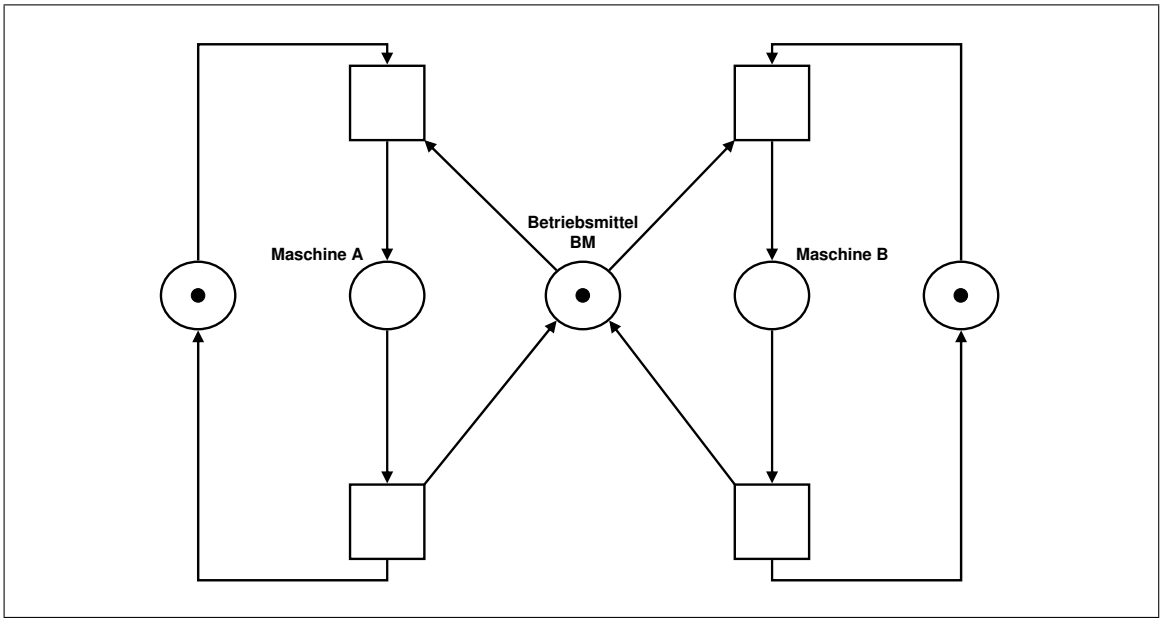


Abb. 5.12: Exemplarisches Petrinetz eines synchronisierten Prozesses

Teileabgabe bereit ist, wird eine Funktion gestartet, welche die nächste Fertigungsstufe und anschließend die zuzuordnende Maschine bestimmt. Im einfachsten Fall könnte der Algorithmus alle in Frage kommenden Ressourcen bezüglich der bereits wartenden Teile überprüfen, und somit jene mit dem geringsten Pufferbestand auswählen. Im Alltag zeigt sich jedoch, dass diese Taktik zu realitätsfremd ist, und dass weitere Parameter berücksichtigt werden müssen. Hierzu zählen Rüstzeiten, die Fertigung von Losen auf mehreren Maschinen parallel oder die Verfügbarkeit von Betriebsmitteln.

Um den Grundsätzen der Simulationsanwendung – der Bewertung und dem Vergleich von Alternativen – gerecht zu werden, empfiehlt es sich die genannte Funktion so zu gestalten, dass diese verschiedene Strategien auswerten kann. Diese alternativen Möglichkeiten in der Realität sind auch die Ursache dafür, dass es wahrscheinlich niemals möglich sein wird, Simulationsmodelle rein aus bestimmten Datenmengen heraus vollständig automatisch zu erzeugen.

Bezüglich der logischen Steuerung haben sich bisher zwei Abstraktionsebenen herauskristallisiert. Auf unterster Ebene steht die Steuerungslogik innerhalb eines Basiselements. Hierzu gehören Funktionalitäten wie das Anfordern eines Werkers, das Abbilden der Beladetechnik, die Bearbeitung selbst, etc.. Es empfiehlt sich diese Logikstufe direkt in den Basiselementen zu verankern, sofern das Simulationssystem dies unterstützt. Dadurch wird insbesondere die Erweiterung um neue Bausteine und auch die Fehlersuche während der Entwicklung vereinfacht. Auf der nächsten Stufe implementiert man die Steuerungslogik innerhalb des Push-Bereichs, also innerhalb des in Abschnitt 5.2.2 beschriebenen Produktionsverbunds. Hier stehen als primäre Anforderungen die Auswahl einer geeigneten Ressource, die Pufferbelegung und die Abbildung des realen Fertigungsablaufs.

Über den beiden genannten Abstraktionsebenen gibt es noch eine übergeordnete Steuerungslogik. Sie stellt einerseits die Kanban-Ebene beim segmentierten ConWIP dar (vgl. Abb. 5.5)

und regelt übergeordnete Anforderungen, wie den Abzug von Teilen durch die Montage, den Zeitpunkt des Einlastens neuer Lose und übergeordnete Pufferbestände zwischen Fertigungsbereichen. Diese Steuerungsebene steuert auch die Quelle und Senke des zu generierenden Simulationsmodells.

Erst das Zusammenspiel aller drei Logikebenen ermöglichen einen korrekten Ablauf des gesamten Modells. Ein gemeinsamer Zugriff auf bestimmte Parameter ist dabei unabdingbar. So muss die Logik des Basiselements wiederum die Losgröße der Modellebene kennen, um entscheiden zu können, ab wann die Maschine auf ein anderes Teil umgerüstet wird. Ebenso muss die Steuerungslogik zur Weitergabe den momentanen Zustand eines Basiselements kennen, um nicht versehentlich Teile auf eine Maschine weiterzugeben, welche gerade repariert wird.

Zur Umsetzung dieser Anforderung kann man auf die Konzepte der Informatik und der Programmiersprachen zurückgreifen. Man unterscheidet hier oftmals zwischen globalen und lokalen Parametern. Die globalen Parameter stehen allen Funktionen und Methoden zur Verfügung, während die lokalen nur innerhalb der jeweiligen Methode sichtbar sind. Man stellt hierdurch auch sicher, dass globale und lokale Steuerungsfunktionen sich nicht gegenseitig behindern bzw. beeinflussen können. Viele Simulationssysteme unterstützen diesen aus der Programmierung bekannten objektorientierten Ansatz. Besteht diese Möglichkeit nicht, wie es bei Quest der Fall ist, so müssen alle Objekt auf einer Ebene sein und die Steuerungslogik wird komplexer, da sie alle Hierarchiestufen selbst erkennen muss. Alle drei Bereiche der Steuerungslogik, wie sie in diesem Abschnitt genannt sind, müssen innerhalb des entsprechenden Simulationssystems umgesetzt werden. Der Vorteil an dem abstrakten Konzept besteht in der Wiederverwendbarkeit und der einheitlichen Parametrisierung, welche bei der Umsetzung des Simulationsdatenframeworks von großer Bedeutung ist. Diese wird in Abschnitt 6.3.3 noch näher bezüglich der Simulationssysteme Quest, Simflex/3D und Automod beschrieben.

Kapitel 6

Umsetzung des Konzepts

In den folgenden Abschnitten soll das in Kapitel 4 beschriebene Konzept zur Implementierung eines Simulationsdatenframeworks und die in Kapitel 5 erläuterten Anpassungen in den Simulationssystemen umgesetzt werden. Dabei soll auf die praktische Realisierung des webbasierten Entwurfs eingegangen werden und es sollen die Möglichkeiten aufgezeigt werden, mit denen bestimmte Daten in eine für die Simulationsanwendung brauchbare Form zu bringen. Die Daten aus verschiedensten, verteilten Datenbanksystemen sollen innerhalb der formalen Beschreibungssprache XML zusammengefasst werden und anschließend zur Generierung eines Simulationsmodells genutzt werden. Anschließend werden die Möglichkeiten und auch die Grenzen dieser Systematik aufgezeigt und Anwendungsgebiete für das Konzept im Planungsalltag beschrieben.

6.1 Webbasierte Umsetzung des entwickelten Konzepts

Im vorangegangenen Kapitel ist eine Abstraktion der realen Fertigung hin zu einem für die Teilefertigung allgemeingültigen Modell durchgeführt worden. Das abstrakte Modell soll in diesem Abschnitt mit den notwendigen simulationsrelevanten Daten versorgt werden. In Abschnitt 4.5 wurde ein Konzept vorgestellt, mit dem es möglich ist, die geforderten Daten aus verschiedenen Systemen bereitzustellen. Der gesamte Umfang an Daten befindet sich innerhalb der Datenbankebene aus Abschnitt 4.5.1. Hinter dieser verbergen sich diverse Produktions- und Planungsdatenbanken der Fabrik. Das Zusammenfügen der Daten aus verschiedenen Elementen der Datenbankebene wurde in Abschnitt 4.5.2 konzeptionell vorgestellt. Diese Middlewareebene des Konzepts soll die zentrale Aufgabe der Datenbereitstellung übernehmen und dabei webbasierte Techniken verwenden. In den folgenden Abschnitten sollen die Implementierung des bereits vorgestellten Konzepts und die notwendigen Anbindungen näher beschrieben werden.

6.1.1 Rahmenbedingungen und Softwareanforderungen

Zur Realisierung des vorgestellten Konzepts sollen die Webtechniken, wie sie heute für den globalen Datenaustausch und für Anwendungen im Internet verwendet werden, zum Einsatz kommen. Daher werden die folgenden Programmmodule in der Programmiersprache Java realisiert, da diese den genannten Anforderungen entspricht und im Internet neben Programmiersprachen, wie C#, Perl, etc., die am weitesten verbreitete und geeignete ist. Die Grundsätze dieser Sprache werden im weiteren Verlauf noch näher erläutert.

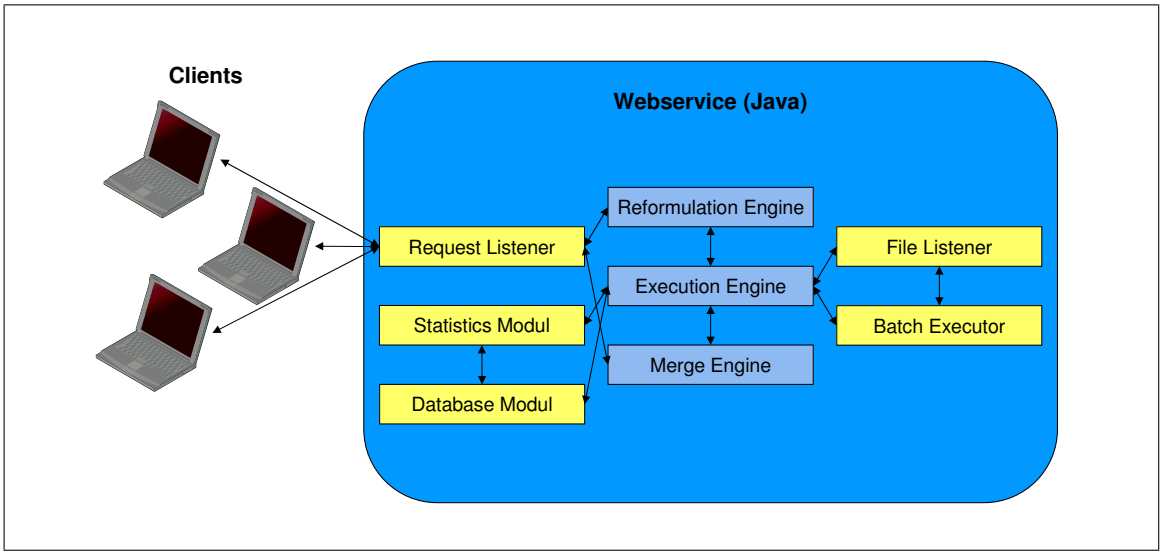


Abb. 6.1: Lose Kopplung von unterschiedlichen Services

Weiterhin soll ein Webserver für die Praxistests eingesetzt werden, welcher die programmierte Logik der Datenzusammenführung beinhaltet. Dieser soll innerhalb des Firmennetzwerks auf Basis des Betriebssystems Windows von Microsoft lauffähig sein. Die Programmierung der im Folgenden beschriebenen Module erfolgte innerhalb der Entwicklungsumgebung JBuilder der Firma Borland. Dabei sind die Module jeweils einzeln ausführbar, da sie teilweise nacheinander programmiert wurden, besitzen jedoch die Fähigkeit mit den anderen Modulen zu kooperieren und somit das Simulationsdatenframework zu bilden.

Die programmiertechnische Umsetzung wurde nach der Service Oriented Architecture (SOA) gestaltet. Diese ist auch unter dem Namen dienstorientierte Architektur bekannt. Der Ansatz kommt aus der verteilten Programmierung, wie sie bei komplexen Geschäftsprozessen bevorzugt angewendet wird. Die Grundidee besteht darin, dass Dienste und Funktionalitäten in Form von Services implementiert werden. Ein Service ist in diesem Zusammenhang eine Funktionalität, welche über eine standardisierte Schnittstelle in Anspruch genommen werden kann. Durch die Aneinanderreihung der einzelnen Services können somit komplexe Anwendungen erstellt werden. Die Programmlogik ist nicht in einem einzelnen Programm zu finden, sondern verteilt über mehrere unabhängige Dienste [Erl04]. Häufig werden für SOAs Web Services eingesetzt [Wik07h].

Abb. 6.1 zeigt den schematischen Aufbau des im folgenden zu implementierenden Frameworks. Die einzelnen Dieste, wie das Statistikmodul oder die Reformulation Engine sind dabei abgeschlossene Softwareanwendungen, welche über eine Schnittstelle einen Dienst anbieten. Dieser kann von den anderen Servicemodulen verwendet werden und bildet in Summe das System zur Generierung von Daten für Simulationsmodelle. Die eingesetzten Verfahren und die Umsetzung der Dienste werden in den folgenden Abschnitten näher beschrieben.

6.1.2 Serverseitige Umsetzung der Middlewareebene

Zentraler Bestandteil bei der Umsetzung des vorgestellten Konzepts ist ein Webserver, welcher die mittlere Ebene der 3-Tier-Anwendung bildet. Dieser muss Verbindungen zu verschiedenen Datenbanksystemen mittels des Verbindungspools, wie es in Abb. 3.6 in Abschnitt 3.3.3 dargestellt ist, verwalten. Zusätzlich muss er die Anfragen von verschiedenen Clients abarbeiten, zu denen letztlich auch Simulationsmodelle gehören werden. Dazwischen stehen noch verschiedene programmiertechnische Umsetzungen zur Zusammenführung, Umsetzung und Auswertung verschiedenster Datenstrukturen.

Der Begriff Webserver kann zwei verschiedene Bedeutungen einnehmen:

1. Ein Rechner, welcher Webseiten (meist HTML) mittels das HTTP-Protokolls Clients (meist Webbrowsern) bereitstellt.
2. Eine Software, welche Webdokumente zur Verfügung stellt.

Die Begrifflichkeit gilt sowohl für den Rechner als Hardware, als auch für die Software, welche den Dienst auf ebendieser Hardware zur Verfügung stellt. Es gibt eine Reihe softwaretechnisch verschiedener Umsetzungen von Webservern, jedoch basieren alle auf derselben Grundlage. Jeder Webserver akzeptiert HTTP-Anfragen über ein Netzwerk und stellt eine HTTP-Antwort für den Anfragenden zur Verfügung. Die HTTP-Antwort besteht typischerweise aus HTML-Dokumenten, kann jedoch auch aus reinem Text, Bildern, XML und anderen Formattypen bestehen. Der historisch erste Webserver wurde am CERN[1] entwickelt, um das dortige, sehr umfangreiche Telefonbuch im Netzwerk bereitzustellen. Die Software lief auf einem NeXT Cube Rechner, welcher ebenfalls zur Entwicklung des ersten Webbrowsers verwendet wurde.

Für die praktische Umsetzung des entwickelten Konzepts soll ein Apache Webserver verwendet werden, welcher zu dem entsprechenden Open-Source-Projekt der Apache Software Foundation gehört. Dieser wird weltweit auf 69.32% der Server eingesetzt[2][Net05]. Dieser kann mittels der Software Jakarta Tomcat, einem ebenfalls offenen Projekt der Apache Software Group, um einen Java-Server-Seiten- und Servlet-Container erweitert werden. Erst mit dieser Erweiterung ist es möglich, auch dynamische Inhalte zur Verfügung zu stellen. Bei der Programmierung muss beachtet werden, dass die Module später auch unter dem Applikationsserver IBM[3]-WebSphere lauffähig sind, da diese für den Produktiveinsatzt aufgrund der kommerziellen Supportleistungen gegenüber den Open-Source-Projekten zu bevorzugen sind.

Das Java-Servlet-API[4] ist heute Stand der Technik im Bereich vieler, kommerzieller Anwendungen rund um das World Wide Web. Hierzu zählen Onlinebanking, Wetterdienste, Suchmaschinen und viele andere Informationsdienste, bei denen es darauf ankommt sich ändernde Daten in eine übersichtliche, standardisierte Darstellung zu bringen. Das Servlet API definiert die erwarteten Interaktionen zwischen Webcontainer und Servlet. Dabei ist der Webcontainer grundsätzlich die Komponente eines Webservers, welcher mit dem Servlet interagiert. Er ist verantwortlich, dass

[1] Conseil Européen pour la Recherche Nucléaire – Europäisches Zentrum für Nuklearforschung
[2] Stand Mai, 2005
[3] International Business Machines
[4] engl. Application Programming Interface

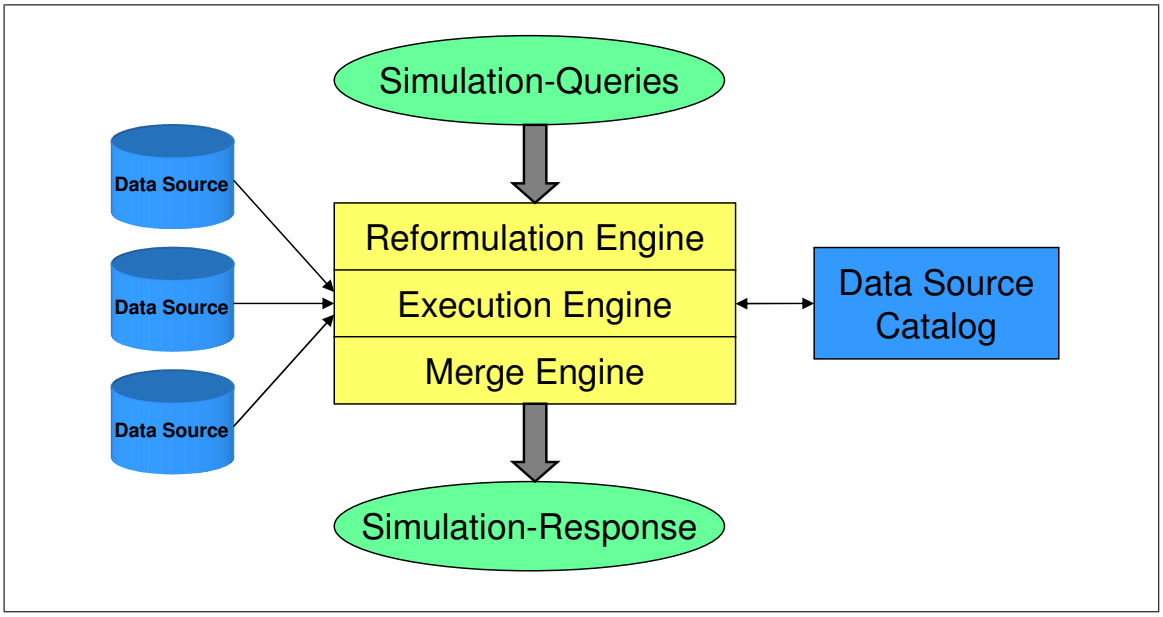


Abb. 6.2: Wesentliche Komponenten der Frameworkarchitektur

Anfragen an den Webserver zu den entsprechenden Servlets weitergeleitet werden, welche dann in Echtzeit die gewünschten Informationen bereitstellen. Die Rückmeldung kann dabei sowohl in Form von HTML, als auch jeder beliebigen anderen Form, wie XML oder Text erfolgen. Diese Tatsache soll für die Umsetzung des Konzepts ausgenutzt werden, so dass letztlich der komplette Dateninhalt eines Simulationsmodells in einer XML-Antwort vom Webserver zurückgeliefert wird.

Von der Datenanfrage bis zur entsprechenden Antwort müssen unterschiedliche Aufgaben innerhalb des Webservers erledigt werden (vgl. Abb. 6.2). Angenommen man benötigt die Simulationsdaten für einen kompletten Fertigungsbereich, so übergibt man die Anfrage an den Server mit einem, oder mehreren Attributen, welche die Identifikation der Daten in anderen Systemen ermöglichen.

Zur Umsetzung des Konzepts müssen verschiedene Servlets als Vermittler und zur Koordinierung der unterschiedlichen Services betrieben werden. Beim Eingang einer Anfrage an den Webserver muss diese in mehrere verschiedene Unterabfragen gesplittet werden, damit diese zu den entsprechenden verteilten Systemen weitergeleitet werden können. Die Rolle des Mediators, wie er in Abschnitt 3.4 und Abb. 3.7 dargestellt ist, besteht folglich aus mehreren Unteraufgaben. Hierzu untergliedert man die Middlewareebene in drei verschiedene Engines (Maschinen) (siehe Abb. 6.2).

6.1.2.1 Die Reformulation Engine

Die Primäre Anfrage an den Webserver beinhaltet Informationen, die zur Identifikation von Daten in den verteilten Systemen dienen. In dem Beispiel aus Abb. 6.3 wäre dies eine Kostenstelle innerhalb der Fabrik. Die Reformulation Engine beginnt folglich die Anfrage in eine erste

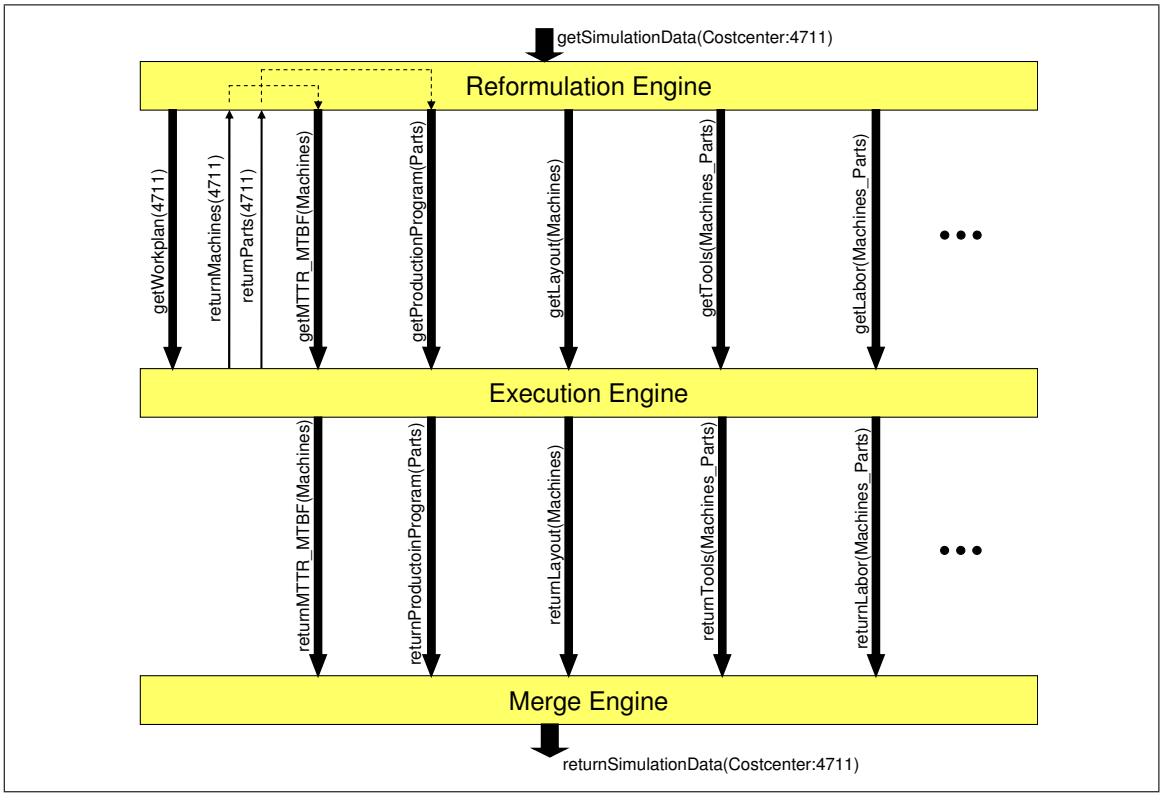


Abb. 6.3: Systematischer Ablauf zwischen den Engines

Unterabfrage zu zerlegen. Dabei werden alle Arbeitspläne innerhalb der Kostenstelle gelesen und die Teilmenge der Maschinen innerhalb dieser zurückgeliefert. Ohne die Bezeichnungen der Maschinen innerhalb einer Kostenstelle zu kennen, wäre es der Reformulation Engine nicht möglich, die Anfrage an ein Instandhaltungssystem weiterzuleiten. Grund dafür ist der Identifikationsschlüssel, nach welchem die Daten in verschiedenen Systemen abgelegt werden. Die Unterabfrage wird zur Ausführung an die Execution Engine übergeben.

Weiterhin werden aus dem Fertigungsplan die entsprechenden Teile, welche innerhalb der Kostenstelle gefertigt werden, zurückgeliefert. Diese müssen dem Leitstands- oder Logistiksystem übergeben werden, damit für diese die Jahresstückzahlen ermittelt werden können. Hierbei muss der Sinn und Zweck einer auszuführenden Simulationsstudie bedacht werden. Je nach Anforderung kann man das annähernd festgeschriebene Produktionsprogramm für die nächsten 8-21 Tage aus dem Leitstand beziehen oder aber die Planzahlen für die nächsten 6 Monate bis 3 Jahre aus den Logistiksystemen anfordern. Diese Anforderung muss bereits der Reformulation Engine vorliegen, damit sie diese entsprechend an die Execution Engine weitergeben kann.

Es ist technisch möglich, das Definieren von Unterabfragen über mehrere Stufen auszuführen. Angenommen es wird als erstes der Fertigungsplan gelesen, um daraus die betroffenen Produkte zu ermitteln. Diese müssen anschließend über ein zweites System nach dem zugehörigen Baumuster auf Einzelteilebene aufgelöst werden. Der Austausch zwischen dem Generieren der Unterabfrage und der Ausführung dieser müsste folglich über drei Stufen erfolgen. Diese Not-

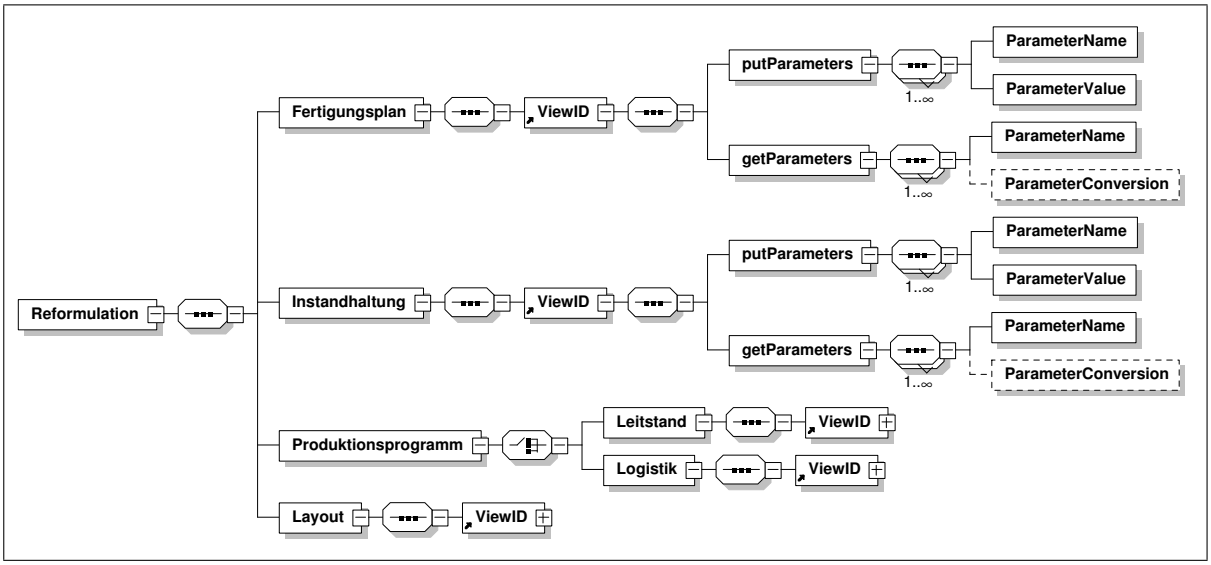


Abb. 6.4: XML-Schema der Reformulation-Konfiguration

wendigkeit besteht jedoch momentan aufgrund der Dokumentationsweise in den betrieblichen Systemen nicht.

In diesem Teil der webbasierten Umsetzung des Konzepts ist auch der erste Teil der in Abschnitt 4.5.5 beschriebenen Konfiguration der Datenzusammenhänge implementiert. Die Ortsbeschreibungen und die Anfragetechnik an die Datenbanken spielen auf dieser Ebene noch keine Rolle. Zur Formulierung von Teilabfragen benötigt man nur die entsprechenden Sichten zu den jeweiligen Systemen. Es muss einerseits die zeitliche Abfolge der Abfragen definiert werden und andererseits müssen die zu übergebenden und zu übernehmenden Parameter festgelegt werden. Das Schema der Reformulation-Konfiguration ist grafisch in Abb. 6.4 dargestellt. Unterhalb des Hauptelements `Reformulation` findet sich bei Einhaltung einer Abarbeitungsfolge von oben nach unten die Reihenfolge der Systeme, die abzufragen sind. Jedem System ist dabei eine eindeutige `ViewID` zugeordnet, welche jedoch erst in der Execution Engine von Bedeutung ist. Zur Ausführung der Abfragezerlegung muss unterhalb des `ViewID` Elements noch definiert sein, welche Parameter bei der Ausführung übergeben werden müssen, und welche für die Formulierung der noch kommenden Abfragen angezogen werden müssen. Hierfür sind die Elemente `getParameters` und `putParameters` mit den jeweiligen Unterelementen `ParameterName`, `ParameterValue` oder `ParameterConversion`. Der Abfrage können laut Definition beliebig viele Parameter übergeben werden, es muss jedoch mindestens einer angegeben sein. Nach erfolgreicher Durchführung werden die unterhalb von `getParameters` angeforderten Felder aus der Datenbank zurückgegeben. Für jeden dieser Parameter kann eine Umwandlungsanweisung definiert werden. Hierfür werden reguläre Ausdrücke verwendet. Diese sind im Schema optional definiert und können bei Bedarf angewendet werden.

Bei der Programmierung der Reformulation Engine wurde eine Initialisierungsroutine implementiert, welche beim Starten des Webservers die oben genannte Konfigurationsinhalte aus einer XML-Datei ließt und zur Ausführung den weiteren Klassen des Moduls im Speicher zur Verfügung stellt. Beim Eintreffen einer Anfrage am Webserver wird diese an eine weitere Klas-

se innerhalb dieses Moduls durchgereicht. Diese nutzt die im Speicher befindliche Konfiguration aus Abb. 6.4, um die Anfrage in Unterabfragen zu zerlegen und diese an die Execution Engine weiterzureichen. Es wurden noch diverse Erweiterungsklassen programmiert, welche eine Parallelisierbarkeit der Anfragen prüft und gegebenenfalls durchführt. Außerdem wartet ein Thread auf die Antworten von der Execution Engine. Die Reformulation Engine ist als abgeschlossenes Modul realisiert, welches eigenständig Anfragen zerlegen kann. Jedoch ist dieses ohne die Antworten von der nachfolgenden Engine nicht nutzbar, da sonst bis zu einem Timeout auf Antworten gewartet wird und anschließend eine Fehlermeldung ausgegeben wird.

6.1.2.2 Die Execution Engine

Nachdem die Hauptabfrage von der Reformulation Engine in die entsprechenden Unterabfragen zerlegt ist, kann die Execution Engine die Daten sofort aus den entsprechenden Datenbanken anfordern und übertragen. Jetzt wird auch der zweite Teil der Konfiguration, wie sie in Abschnitt 4.5.5 als ganzes beschrieben ist, benötigt. Hierzu gehören die physikalischen Adressen der entsprechenden Datenbanken, der Instanzname und die zu benutzende Technik zur Datenanfrage (SQL, Batch, File-Exchange, etc.). Außerdem müssen die simulationsrelevanten Felder definiert und die entsprechenden Schlüsselfelder zwischen unterschiedlichen Datenbanken beschrieben sein, damit die Zusammenführungstechnik aus Abschnitt 4.5.2.1 angewendet werden kann.

Die Konfiguration der Zusammenführungslogik ist komplex strukturiert und beschreibt den in Abb. 6.2 genannten „Datasource Catalog“. Wie bereits in Abschnitt 4.5.5 beschrieben, müssen Abfragen rekursiv über mehrere Ebenen und Datenbanksysteme möglich sein. Das Schema zur Darstellung dieser Zusammenhänge beginnt mit dem Hauptelement der Sichten (`Views`) (vgl. Abb. 6.5). Darunter müssen letztlich alle einzelnen Sichten auf die Datenbanksysteme aufgeführt und inklusive der Referenzen auf andere Datenbanken definiert sein. Zu jeder Sicht gehört die physikalische Beschreibung des zugehörigen Datenbankzugriffs in Form einer Netzwerkadresse (`IP-Address`), einer Portnummer (`Port`) und einem Instanznamen (`Instance`). Zusätzlich wird noch ein Element für die Implementierung zur Verfügung gestellt. Dieses ist innerhalb des Programmcodes notwendig, um den richtigen Datenbanktreiber (z.B. Oracle, IBM DB2, MS-SQL, etc.) zu wählen, oder bei Dateiformen und Batchverarbeitung die entsprechende Programmroutine zur Umwandlung zu starten. Die Zugriffsbeschreibungen kommen jeweils nur einmal vor, während die Tabellen (`Table`), welche die simulationsrelevanten Daten enthalten, in der Anzahl beliebig sein können ($1 \ldots \infty$).

Zur Umsetzung der rekursiven Tabellenabfragen über mehrere Systeme, muss die Konfiguration auch eine rekursive Beschreibung zulassen. Dies wird mit Hilfe einer Schemareferenz innerhalb des XML-Schemas erreicht. Unterhalb der Tabelle befinden sich die anzufragenden Felder, welche wiederum eine Referenz auf eine andere Sicht (`ViewID`) haben können. Dabei wird auch das Schlüsselfeld (`FieldName`) und die Art und Weise der Referenzierung (`ReferenceType`) festgelegt. Die Referenzierungsart ist an die Datenbankanfragesprache SQL angelehnt und kann die darin enthaltenen Verknüpfungstypen (Joins) definieren. Dadurch wird es möglich, alle Merkmale aus Tabelle A einzubeziehen und nur diejenigen aus Tabelle B, welche im Schlüsselfeld übereinstimmen (Left Join). Damit ist eine rekursive Sichtendefinition

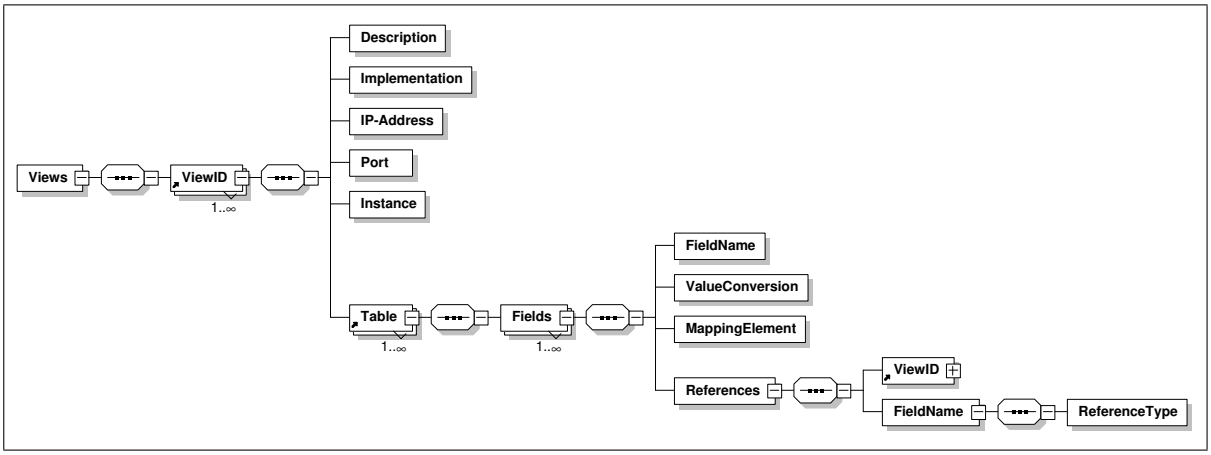


Abb. 6.5: XML-Schema der Execution-Konfiguration

erfüllt und die Execution Engine kann jede beliebe Abfrage ausführen, wenn der entsprechende Name und die Parameter übergeben werden. Zusätzlich kann jedes einzelne Feld noch einer Umformung unterzogen werden (`FieldConversion`). Diese wird wieder über reguläre Ausdrücke definiert.

In Bezug auf die Programmierung ist diese Engine die aufwändigste. Diese besteht einerseits aus Klassen, welche die Kommunikation mit unterschiedlichen Datenbanksystemen realisieren und andererseits der Logik zur rekursiven Ausführung von Unterabfragen und deren Zusammenführung im Hauptelement. Die Klasse zur Ausführung von Datenbankverbindungen liefert der Logik ein Objekt (Data Source), welches eine direkte Kommunikationsverbindung zu einer Datenbank repräsentiert. Auf diesem Objekt kann direkt eine SQL-Anfrage ausgeführt und die Ergebnismenge durchlaufen werden. Die Klassen zur Ausführung der Anwendungslogik nutzen mehrere dieser Data Sources parallel, um die bereits beschriebene, rekursive Anfrageausführung durchzuführen. Dabei wird die in Abschnitt 4.5.2.1 definierte Zusammenführungslogik programmiertechnisch umgesetzt.

6.1.2.3 Die Merge Engine

Der Merge- oder auch Zusammenführungsmechanismus hat die Aufgabe, die ausgeführten Abfragen wieder in eine Gesamtstruktur zu überführen. Diese Anforderung lässt sich mit XML sehr unkompliziert realisieren. Einerseits kennt dieser die Anfragekonfiguration der Reformulation Engine aus Abschnitt 6.1.2.1 und andererseits muss dieser nur die Einzelergebnisse der Execution Engine aus Abschnitt 6.1.2.2 unterhalb eines neuen Hauptelements vereinigen. Die Datenbankfelder weisen teilweise kuriose Feldnamen auf, welche nicht den Ansprüchen von XML genügen, eine beschreibende Syntax zu liefern. Daher ist in der Definition der Sichten aus Abb. 6.5 für jeden Feldnamen der Datenbank ein Element enthalten, welches die Zusammenführung der Merge Engine innerhalb einer neuen Struktur definiert (vgl. Abb. 6.6).

Die unstrukturierten Daten aus der Datenbank bilden eine Liste aus der Kombination Feldname und Feldinhalt. Weitergehende Strukturen stellen relationale Datenbanksysteme nicht zur

```
<Maschine>
    <BANF>000J00210061</BANF>
    <Inv-Nr>J00210061</Inv-Nr>
    <Benennung>Entgratmaschine</Benennung>
    <Positionierung>
        <Pos-X>33.138</Pos-X>
        <Pos-Y>67.723</Pos-Y>
        <Ausrichtung>90</Ausrichtung>
    </Positionierung>
    <Ausfaelle>
        <MTBF>
            <Verteilung>
                <Normal>
                    <my>522.19</my>
                    <sigma2>179438.05</sigma2>
                </Normal>
            </Verteilung>
        </MTBF>
        <MTTR>
            <Exponential>
                <lambda>2.41</lambda>
            </Exponential>
        </MTTR>
    </Ausfaelle>
    <Lader>Hand</Lader>
</Maschine>
```

Abb. 6.6: Ergebnis nach Durchlauf aller Engines (Auszug)

Verfügung. Zur Generierung von Simulationsmodellen ist es jedoch hilfreich, eine strukturierte Form der Daten zu haben, wie sie schon in SDX eingeführt wurde. Der Inhalt des in der Definition genannten Elements `MappingElement` legt folglich fest, wie das Datenbankfeld in der Simulationsstruktur heißen soll und an welcher Stelle es in der Struktur steht.

Innerhalb der Klassen im Programmcode wird in einem ersten Schritt ein leeres XML-Dokument für die Antwort auf die Anfrage im Speicher erstellt. Dieses wird sukzessive mit den Ergebnissen der Execution Engine befüllt und nach dem Eintreffen des letzten Ergebnisses dem Webserver zur Antwort an den Client zur Verfügung gestellt.

6.1.3 Realisierung der Datenbankzugriffe

Stand der Technik in der Anbindung von Datenbanken an Webservices ist das in Abschnitt 3.3.3 und Abb. 3.6 dargestellte Connection Pooling. Dabei wird eine definierte Anzahl von gleichzeitigen Verbindungen zu einem Datenbanksystem aufgebaut, welche dann zur parallelen Abarbeitung von Anfragen verwendet werden können. In den heutigen Fabriken existieren jedoch auch noch diverse Hostsysteme, welche teilweise nur eine sogenannte Batchverarbeitung zur Verfügung stellen. Auch nächtliche Exporte von bestimmten Daten auf Netzlaufwerke und die zugehörigen Deltadateien sind durchaus noch gängig. Um diese verschiedenartigen Datengewinnungstechniken innerhalb der Execution Engine aus Abschnitt 6.1.2.2 anwenden zu können, müssen diese programmiertechnisch implementiert werden. In Abb. 6.5 ist das Schemaelement `Implementation` für die Auswahl der entsprechenden Implementierung verantwortlich.

6.1.3.1 Implementierung eines Connection-Pools

Bei der Umsetzung des Simulationsdatenframeworks kommen die Techniken der Webservices zum Einsatz. Diese bauen alle auf der Programmiersprache Java auf. Java ist eine objektorientierte, plattformunabhängige Programmiersprache und als solche ein eingetragenes Warenzeichen der Firma Sun Microsystems. Üblicherweise benötigen Java-Programme zur Ausführung eine spezielle Umgebung (Java Virtual Machine). Der Vorteil ist, dass nur diese Umgebung an verschiedene Computer und Betriebssysteme angepasst werden muss. Sobald dies geschehen ist, laufen auf der Plattform alle Java-Programme ohne Anpassungsaufwand. Hauptziel der Java-Entwicklung war die Internetprogrammierung und Netzwerktechnik, in der die Sprache bis heute zum Stand der Technik zählt [Wik07e].

Zur Verbindung zwischen der Programmiersprache und einem Datenbanksystem wurde JDBC (engl. Java Database Connectivity) entwickelt. Das JDBC-API ist eine einheitliche Schnittstelle zu Datenbanken verschiedener Hersteller, die speziell auf relationale Datenbanken ausgerichtet ist. Es ist in seiner Funktion als Universalschnittstelle vergleichbar mit ODBC unter Windows oder DBI unter Perl. Zu den Aufgaben von JDBC gehört es, Datenbankverbindungen aufzubauen und zu verwalten, SQL-Anfragen an die Datenbank weiterzuleiten und die Ergebnisse in eine für Java nutzbare Form umzuwandeln und dem Programm zur Verfügung zu stellen [Wik07c]. Die Datenbanktreiber, welche für die Verbindung zu unterschiedlichen Produkten (z.B. IBM DB2, Oracle, etc.) benötigt werden, existieren in unterschiedlichen Formen. Man unterscheidet die vollständig in Java umgesetzten (Pure Java) und die nur teilweise in Java umgesetzten Treiber (Partial Java). Während die Pure-Java-Treiber über verschiedene Betriebssystemplattformen hinweg ohne weitere Installationsschritte verwendet werden können, basieren die Partial-Java-Treiber meist auf betriebssystemabhängigen Installationen, wie Dynamic-Link-Libraries (DLL) auf Windows. Im Rahmen dieser Arbeit werden beide Arten von Treibern genutzt, da die Hersteller von Datenbanksoftware meist Partial-Java-Treiber kostenlos zur Verfügung stellen. Die oftmals robusteren und technisch ausgereifteren Pure-Java-Treiber werden in der Regel über Softwareanbieter verkauft und lizenziert und sind im Rahmen dieser Arbeit zur Anbindung an Microsoft SQL-Server im Einsatz.

Die Execution Engine aus Abschnitt 6.1.2.2 dient zur Ausführung der in Abb. 6.5 konfigurierten Abfragen in SQL. In Abschnitt 3.3.3 sind die Vorzüge bzgl. Leistung und Geschwindigkeit einer Umsetzung mittels eines Connection-Pools beschrieben. Das Simulationsdatenframework soll in der Lage sein, mehrere Client-Anfragen simultan zu bearbeiten und zu beantworten. Daher ist es naheliegend, die Verwaltung eines Connection-Pools nicht innerhalb der Execution Engine zu realisiere, sondern dieses übergeordnet dem Server zu überlassen. Diese Technik ist in den meisten offenen und auch kommerziellen Webservern bereits implementiert (z.B. Apache Tomcat, IBM Websphere, etc.) und kann von jedem beliebigen Rechner aus über Intra-/Internet angepasst werden.

Eine Applikation muss nur noch den jeweils zugeordneten Namen einer Datenquelle kennen (z.B. jdbc/InventoryDB). Dieser Bezeichner wird über das JNDI[5] serverseitig aufgelöst. JNDI

[5] engl. Java Naming and Directory Interface

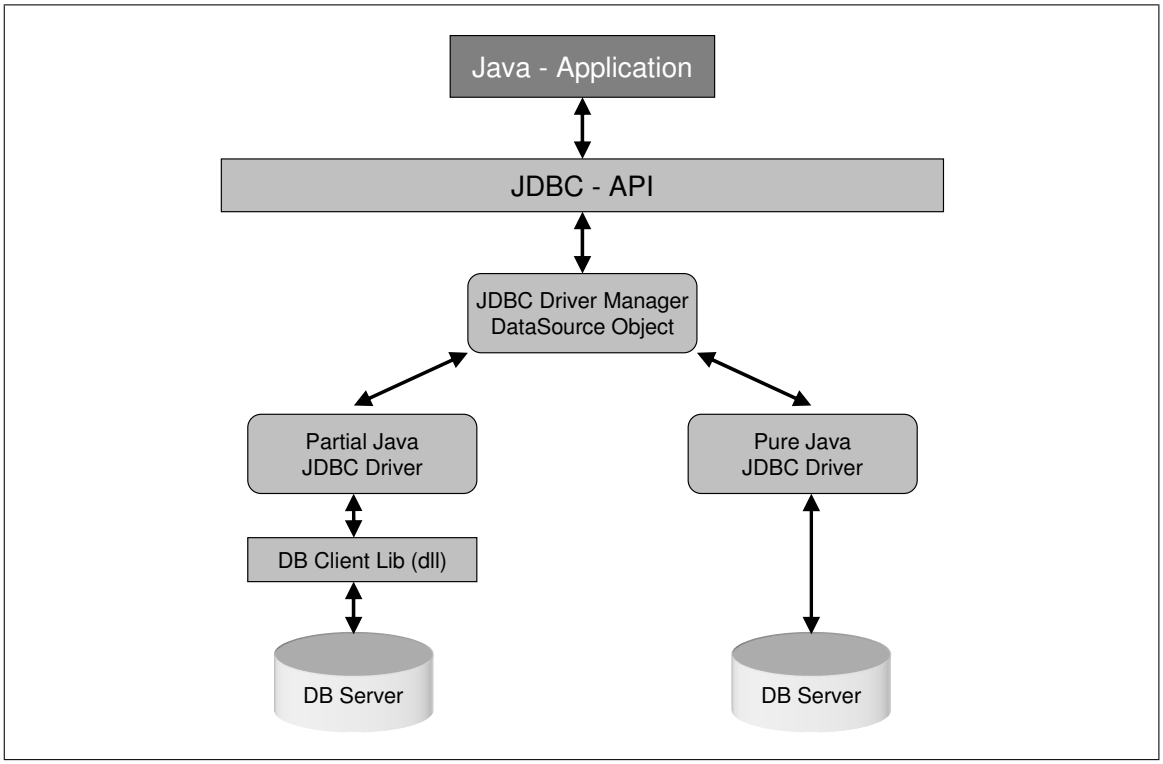


Abb. 6.7: Schematische Darstellung der JDBC-Technik

ist eine Programmierschnittstelle (API) innerhalb der Programmiersprache Java für Namensdienste und Verzeichnisdienste. Mithilfe dieser Schnittstelle können Daten und Objektreferenzen anhand eines Namens abgelegt und von Nutzern der Schnittstelle abgerufen werden. Die Schnittstelle ist dabei unabhängig von der tatsächlichen Implementierung [Wik07d]. In Abb. 6.8 ist der schematische Ablauf eines Verbindungsaufbaus dargestellt. Das JNDI stellt ein Objekt vom Typ DataSource zur Verfügung, welches dann von der Execution Engine genutzt werden kann. Mit dem Aufruf der Methode `getConnection()` wird aus dem Pool der Verbindungen eine zur Verfügung gestellt, welche dann für andere Anfragen gesperrt ist. Sobald die Anfrage vollständig ausgeführt ist, kommt die Verbindung zurück in das Pool der verfügbaren Verbindungen. Der Lifecycle einer Datenbankverbindung ist in Abschnitt 3.3.3 schematisch in Abb. 3.6 dargestellt.

Die Implementierung der jeweiligen Verbindung ist abhängig vom Typ des Anzufragenden Datenbanksystems. Der zu verwendende Treiber ist dem Server bekannt und wird der aufrufenden Applikation anhand des Objekts vom Typ DataSource schon korrekt übermittelt. Der große Vorteil dieser Anwendung, die heute Stand der Technik ist, liegt in der Unabhängigkeit des Programmcodes von der Datenquelle. Angenommen eine Datenbank wird auf einen anderen Rechner umgezogen oder es wird auf einen anderen Datenbankhersteller umgestellt, so muss keine einzige Programmzeile angepasst werden. Innerhalb der Anwendung ist nur der Name (z.B. jdbc/InventoryDB) festgelegt. Der Ort der Datenbank, die zu verwendende Implementierung und auch die Anzahl der bereitzustellenden Verbindungen kann online auf dem Server umgestellt werden oder mit der Konfiguration aus Abb. 6.5 eingespielt werden.

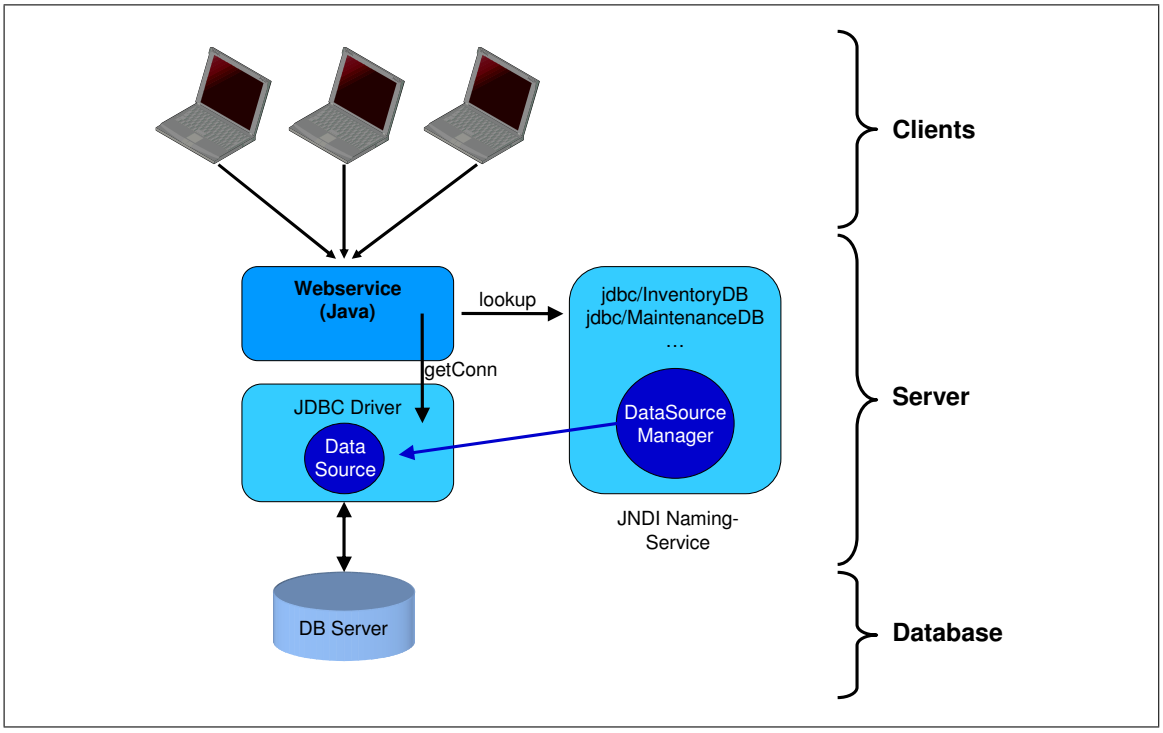


Abb. 6.8: Schematische Darstellung einer DataSource-Verbindung

6.1.3.2 Implementierung von Batch- und Dateizugriffen

In Abschnitt 3.3 wurde bereits erwähnt, dass es nicht immer möglich ist, auf alle produktionsrelevanten Systeme durch eine permanente Verbindung mittels der Connection-Pool-Technik zuzugreifen. Mögliche Ursachen hierfür sind produktionskritische Systeme (z.B. Montagesteuerung), Datenbanken, welche in der Hoheit von externen Betreibern liegen und sehr komplexe Systeme, die ohne Zusatzprogrammierung keine Zugriffe auf die unterliegende Datenbank ermöglichen. In den Abschnitten 3.3.1 und 3.3.2 werden mögliche Varianten aufgezeigt, welche einen annähernden Sofortzugriff erlauben, ohne direkte Anfragen an die Datenbanken zu stellen. Eine in der Industrie häufig vorkommende Variante sind Dateiabzüge und Deltadateien. Abzüge von bestimmten Datenumfängen werden in der Industrie häufig dann eingerichtet, wenn es darum geht mehrere Systeme regelmäßig mit Daten von stammdatenführenden Systemen zu versorgen. Insbesondere im Bereich der Integration von SAP-Softwareprodukten, welche heute nahezu in jedem mittelständigen und größeren Industriebetrieb eingesetzt werden [Wik07g], trifft man häufig auf diese Art der Datenversorgung.

Die Exportdateien weisen unterschiedliche Formate auf. Zu den gängigsten zählen die durch Semikola oder feste Spaltenbreite getrennten Dateien. Dabei repräsentiert jede Zeile innerhalb der Datei einen Datensatz und jede Spalte ein Feld der Datenbank. Noch sehr ungebräuchlich, jedoch mit steigender Tendenz bzgl. des Auftretens, findet man auch schon strukturierte Exporte auf Basis von XML. Hinter dem Begriff der Deltadatei verbirgt sich ein Ausschnitt aus der gesamten Datenmenge, welcher sich seit dem letzten Abzug verändert hat. Es gibt eine Datei mit einer Gesamtmenge der Daten und täglich, wöchentlich, etc. eine Deltadatei, welche nur noch

die seitdem veränderten Zeilen enthält. Diese Änderungslisten beinhalten noch ein zusätzliches Merkmal, welches für jede Zeile besagt, ob diese verändert, hinzugefügt oder gelöscht wurde.

Technisch stehen für die Implementierung solcher Daten in ein Simulationsdatenframework mehrere Möglichkeiten zur Verfügung:

1. Regelmäßiger Import in eine Datenbank und Verfahren nach Abschnitt 6.1.3.1.
2. Halten der Daten im Arbeitsspeicher des Servers.
3. Umwandlung in strukturierte XML-Dateien und lesen bei Bedarf.
4. Lesen der Textdatei bei jeder Anfrage.

Legt man großen Wert auf die Anfragegeschwindigkeit, so ist die automatische Versorgung der Textdateien in eine Datenbank (1) die effizienteste Möglichkeit. Sie sollte beispielsweise bei simulationsbasierten Frühwarnsystemen gewählt werden, bei der ein Simulationssystem sehr schnell auf geänderte Daten der Realität reagieren soll. Geht man jedoch davon aus, dass die Datenmenge klein ist und in annähernd jeder Anfrage benötigt wird, so kann man auch den Weg über die Ablage im Hauptspeicher (2) des Servers wählen. Diese Methodik würde bei der Speicherung der Jahresstückzahlen von einer überschaubaren Menge an Teilen Vorteile bringen. Handelt es sich jedoch um Daten, welche nur sehr selten benötigt werden, so empfiehlt sich das direkte Lesen der Datei bei Bedarf (3 oder 4).

Die einfachste Art der Implementierung besteht in der Überführung der Textdateien in eine eigens dafür eingerichtete Datenbank. Angenommen eine Vertriebsdatenbank gibt jede Nacht eine Datei mit einer Teilnummer und der Absatzschätzung für die nächsten sechs Monate auf ein Netzlaufwerk aus. In einem ersten Schritt richtet man eine Datenbank mit einer Tabelle ein, welche aus den Feldern Teilenummer und Stückzahl besteht. Diese wird wie jede andere Datenbank auch auf Basis der Konfiguration aus Abb. 6.5 in das Simulationsdatenframework aufgenommen und in das Connection-Pool integriert. Nun wird auf dem Server eine Hintergrundanwendung implementiert, welche in regelmäßigen Zeitabständen überprüft, ob sich die Datei auf dem Netzlaufwerk verändert hat. Dieser File-Listener ist ein einfacher Thread, welcher sich jede Stunde oder nach einem festen Zeitintervall einmal aktiviert und in der restlichen Zeit keinerlei Ressourcen verbraucht. Liegt eine veränderte Datei vor, so wird eine Routine aufgerufen, welche die Textdatei einliest und Zeile für Zeile über die Execution Engine in die Datenbank schreibt (vgl. Abb. 6.9). Anschließend wird der File-Listener wieder für die vorbestimmte Zeit deaktiviert. In der Datenbank befinden sich nun die aktualisierten Daten.

Ähnlich verhält es sich mit den Batchverarbeitungen aus Abschnitt 3.3.2. Sie funktionieren ähnlich wie SQL-Anweisungen auf einer direkten Datenbankschnittstelle. Jedoch wird hierbei eine vom jeweiligen Batchsystem lesbare Anfrage auf ein Netzlaufwerk gelegt. Von dort arbeitet das Batchsystem der Reihe nach die Anfragen ab und stellt diese auch auf einem Netzlaufwerk wieder zur Verfügung. Dieses „Quasi-SQL“ hat den großen Nachteil, dass es sich nicht in ein Connection-Pool integrieren lässt und somit auch mehr Rechnerressourcen benötigt. Die SQL-Abfrage der Reformulation Engine muss umgewandelt werden für die Batchverarbeitung. Anschließend muss die Aufgabe dem System übergeben werden und ein File-Listener muss warten,

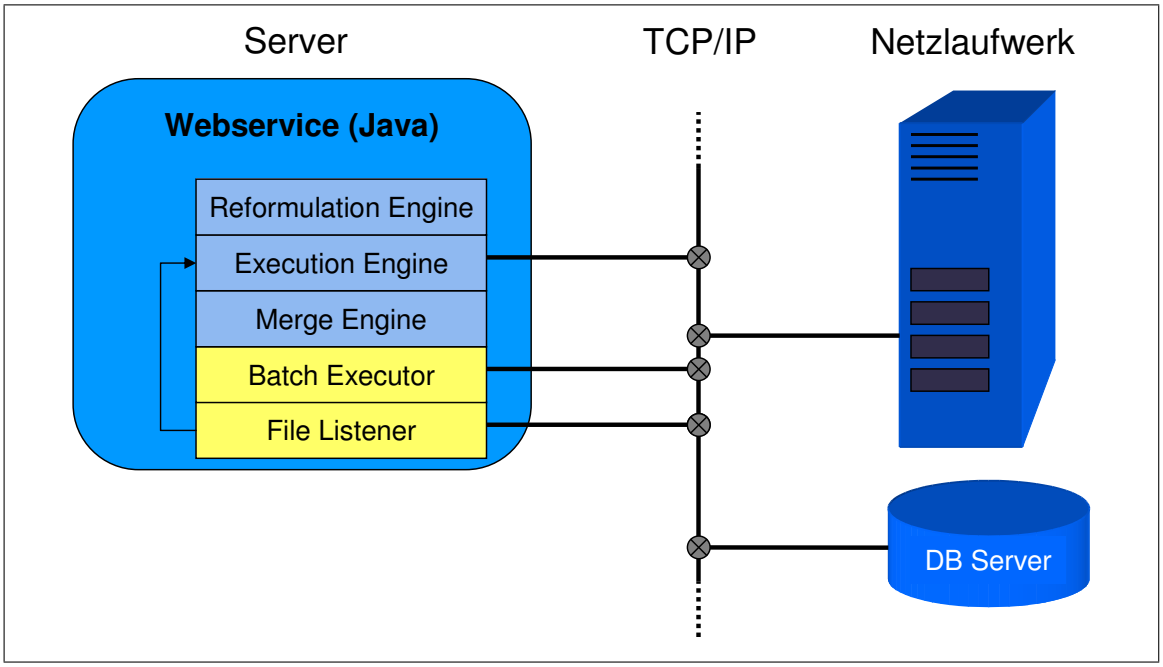


Abb. 6.9: Schematischer Aufbau des Datenframeworks

bis die Ergebnisdatei zur Verfügung steht. Diese muss in den Speicher eingelesen und anschließend der Execution Engine übergeben werden. Ab dann kann nach dem bereits beschriebenen Konzept fortgefahren werden.

In der Praxis hat sich gezeigt, dass die Implementierung der Batchverarbeitung in das Simulationsframework einige Nachteile und auch Risiken mit sich bringt. Einerseits werden hohe Anforderungen an die Rechnerressourcen gestellt und andererseits warten alle in Abb. 6.3 dargestellten, nachfolgenden Anfragen auf das Ergebnis. Bei sehr großen Antwortdateien besteht das Risiko, dass der Hauptspeicher nicht ausreicht und bei langen Verarbeitungszeiten des Batch-Jobs können Zeitüberschreitungen im Netzwerk auftreten, so dass eine gesamte Simulationsanfrage mit einem Fehler beendet wird. Es empfiehlt sich daher diese beschriebene Vorgehensweise, so weit es möglich ist, zu vermeiden.

6.2 Generieren simulationsrelevanter Daten

Über das bereits entwickelte Simulationsdatenframework ist es möglich, eine Vielzahl von Daten über die Produktion und die dafür benötigten Ressourcen in Echtzeit zu erhalten. Nicht alle Daten sind jedoch direkt aus den Produktionssystemen für die Simulation zugänglich bzw. nutzbar. Ein Simulationssystem kann nichts mit der Liste der Ausfalldaten einer bestimmten Ressource anfangen. Es ergibt sich hieraus die Anforderung, dass bestimmte Daten mit mathematischen und statistischen Verfahren in ein für die Simulation verständliches Format gebracht werden müssen.

```
<?xml version="1.0" encoding="UTF-8"?>
<Instandhaltung>
    <Ausfaelle invnr="210061">
        <Ausfall>
            <Datum>2004-11-14 16:58</Datum>
            <Dauer>3.52</Dauer>
        </Ausfall>
        <Ausfall>
            <Datum>2004-11-15 11:40</Datum>
            <Dauer>0.35</Dauer>
        </Ausfall>
        <Ausfall>
            <Datum>2004-11-17 03:42</Datum>
            <Dauer>1.25</Dauer>
        </Ausfall>
        <Ausfall>
            <Datum>2004-11-19 09:13</Datum>
            <Dauer>3.57</Dauer>
        </Ausfall>
        <Ausfall>
            <Datum>2004-11-22 18:11</Datum>
            <Dauer>2.51</Dauer>
        </Ausfall>
        <Ausfall>
            <Datum>2004-11-23 03:40</Datum>
            <Dauer>0.50</Dauer>
        </Ausfall>
        <Ausfall>
            <Datum>2004-11-25 15:31</Datum>
            <Dauer>1.75</Dauer>
        </Ausfall>
    </Ausfaelle>
</Instandhaltung>
```

Abb. 6.10: Darstellung von Maschinenausfällen in XML

Hierfür wurde ebenfalls ein Modul in Java programmiert, welches auf dem Webserver lauffähig ist. Dieses Modul ist in der Lage, beliebige Merkmalsausprägungen bezüglich statistischen Verteilungen zu untersuchen. Der Aufbau des Moduls wurde in drei Stufen durchgeführt. Eine Javaklasse nimmt die zu analysierenden Daten auf und stellt notwendige Methoden zur Verfügung, welche es erlauben Mittelwerte, Standardabweichungen und weitere Kennzahlen zu berechnen. Auf dieser Klasse aufbauend wurden dann die in den folgenden Abschnitten beschriebenen statistischen Testverfahren programmiert, welche es ermöglichen aus den Ursprungsdaten statistische Verteilungen abzuleiten. Zusätzlich besteht die Möglichkeit eine empirische Verteilungsfunktion aus den Eingangsdaten zu erstellen. Die mathematischen Grundlagen, auf welchen das Programmmodul basiert, sollen im Folgenden näher beschrieben werden.

6.2.1 Statistische Verfahren zur Datenumwandlung

Zur Beschreibung einer Maschine mit den zugehörigen, simulationsrelevanten Daten gehören bestimmte Parameter, wie sie bereits in Abb. 4.10 dargestellt sind. Dieses XML-Schema ist in Abb. 6.6 exemplarisch mit Daten aus den verteilten Produktionssystemen gefüllt. Das Simulationssystem arbeitet mit statistischen Merkmalsausprägungen bei der zeitlichen Einplanung von Maschinenausfällen und -reparaturen. Aus den Produktionssystemen erhält man jedoch eine diskrete Darstellung der Instandhaltungsdaten (vgl. Abb. 6.10 und Abb. 6.11). Diese Informationen sollen umgewandelt in eine statistische Verteilungsfunktion in die Maschinenbeschreibung aus Abb. 6.6 einfließen.

Aus diesen diskreten Merkmalen gilt es nun zwei statistische Größen abzuleiten – die MTBF und die MTTR. Die MTBF legt die statistische Verteilung der Abstände zwischen zwei Maschinenausfällen fest, während die MTTR die Verteilung der Reparaturdauer bestimmt. Der Zusammenhang zwischen der Verfügbarkeit einer Ressource A_D und der MTBF oder MTTR ist in (6.1) beschrieben.

$$A_D = \frac{MTBF}{MTBF + MTTR} \tag{6.1}$$

In der Fachliteratur werden häufig Annahmen über Verfügbarkeiten und Reparaturdauern von Maschinen und Anlagen getroffen. So wird bei hochautomatisierten Maschinen häufig die geometrische Verteilung von Dallery und Gershwin angenommen [DG92]. Durch die Anbindung an die Produktionssysteme steht jedoch eine vollständige Datenreihe aus dem Instandhaltungssystem für eine zu simulierende Ressource zur Verfügung. Man nennt diese Datenreihe auch Urliste $x_1, x_2, ..., x_n$. Diese Liste beinhaltet die Menge aller durch eine Datenerhebung gewonnenen Daten bzgl. eines oder mehrerer Merkmale. In dem genannten Beispiel gewinnt man aus einer Urliste zwei für die Simulation interessante Merkmalsausprägungen:

- Die Zeiten zwischen den Ereignissen (MTBF)
- Die Zeitdauer der jeweiligen Ereignisse (MTTR)

Die Merkmalsausprägungen der Werte aus einer Urliste werden mit $a_1, a_2, ..., a_k;\ k \leq n$ benannt.

Die Anzahl der zur Verfügung stehenden Daten in den Produktionsdatenbanken können stark schwanken. Um das Verfahren zur Generierung simulationsrelevanter Daten möglichst flexibel zu gestalten, werden im Folgenden zwei statistische Anpassungstests gewählt, welche sowohl große als auch kleine Datenmengen auf Zugehörigkeit zu Verteilungen testen können. Die Tests dienen dabei lediglich zur Überprüfung einer Hypothese über die Verteilungsfunktion einer Grundgesamtheit. Es kann lediglich bewiesen werden, dass die Hypothese nicht falsch ist, nicht jedoch die Richtigkeit einer Annahme [Sac97].

6.2.1.1 Chi-Quadrat-Anpassungstest

Dieses Prüfverfahren ermöglicht es, bei großen Stichprobenumfängen oder einer großen Anzahl von Merkmalsausprägungen, eine Zugehörigkeit zu einer statistischen Verteilung zu testen. Am Beispiel einer Normalverteilung stellt man die Hypothese

$$H_0 : \text{die Grundgesamtheit ist } N(\mu_0, \sigma_0^2) \text{ verteilt}$$

gegen die Alternative

$$H_1 : \text{die Grundgesamtheit ist nicht } N(\mu_0, \sigma_0^2) \text{ verteilt}$$

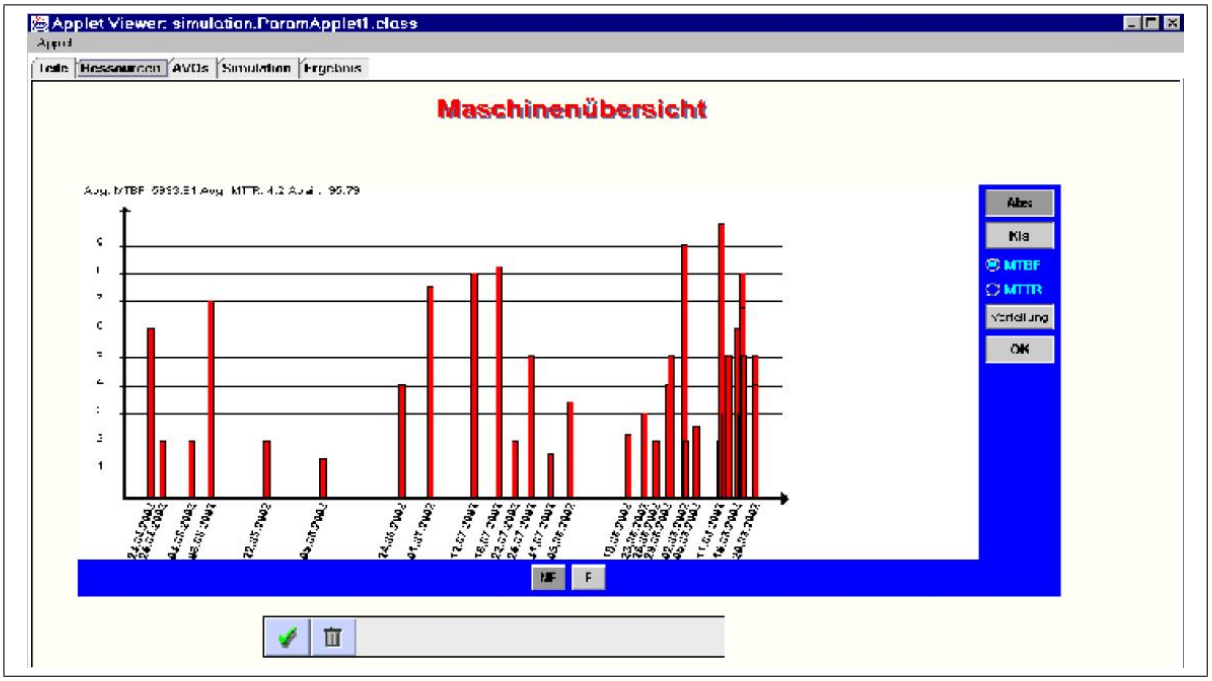


Abb. 6.11: Exemplarische Darstellung MTBF und MTTR

auf. Dabei kann man einerseits den Fehler machen, dass man sich für H_1 entscheidet, obwohl H_0 vorliegt. Man nennt dies den Fehler 1. Art. Andererseits kann man sich auch fälschlicherweise für H_0 entscheiden. In diesem Fall spricht man von einem Fehler 2. Art. In einer konkreten Situation weiß man natürlich nicht, ob man einen Fehler macht, sondern lediglich welcher Art dieser ist. Ist jedoch bekannt, dass das verwendete Entscheidungsverfahren nur mit einer Wahrscheinlichkeit von höchstens α den Fehler 1. Art (α-Fehler) macht, so spricht man von einem Test zum Niveau α. Daher wird α auch als Signifikanzniveau bezeichnet. [Har98]

Aufgrund der großen Datenmengen werden die Daten in Klassen $K_1, K_2, ..., K_p$; $p \leq n$ eingeteilt, welche durchschnittsfremde Intervallbereiche repräsentieren. Als Faustformel für die Anzahl der Klassen p gilt:

$$p \approx 5 * \ln n \tag{6.2}$$

In einem weiteren Schritt werden die gesamten Merkmalsausprägungen von x den Klassen $K_1, ..., K_p$ zugeordnet, so dass jedes nur denkbare Merkmal x in genau eine Klasse K_i fällt ($x \in K_i$). Man erhält die absoluten Klassenhäufigkeiten $O_i, i = 1, ..., p$. Falls Klassenhäufigkeiten zu klein sind ($n < 5$), werden benachbarte Klassen zusammengelegt.

Es muss anschließend festgelegt werden, auf welche Verteilung die Hypothesen getestet werden sollen. Um bei dem Beispiel der Normalverteilung zu bleiben, gilt es die Parameter μ und σ^2, welche eine Normalverteilung bestimmen, festzulegen. Hierfür kann man einerseits die Maximum-Likelihood-Methode aus den Klassenhäufigkeiten benutzen oder aber den Stichprobenschätzer $m = \bar{x}$ und $s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$ verwenden [Bol04]. Bei der Verwendung der Stichprobenschätzer verringert sich die Anzahl der Freiheitsgrade um 2. Dies wirkt sich bei

der Ermittlung der Annahme- oder Ablehnungsschwelle aus. Die Annahme der Hypothese fällt daher eher konservativ aus [CL54].

Zur Berechnung der Prüfgröße nach (6.3), welche dann entscheidend für die Annahme oder Ablehnung der Hypothese H_0 ist, benötigt man die oberen und unteren Klassengrenzen k_{i_o} und k_{i_u}.

$$T = \sum_{i=1}^{k} \frac{1}{E_i}(O_i - E_i)^2 \tag{6.3}$$

Setzt man für die erwartete Klassenhäufigkeit E_i bei einer Normalverteilung die Differenz der Verteilungsfunktion $\Phi(x;\bar{x};\sigma^2)$ an der oberen und unteren Klassengrenze multipliziert mit der Anzahl der Stichproben n in (6.3) ein, so erhält man:

$$T = \sum_{i=1}^{p} \frac{((\Phi(k_{i_o};\bar{x};\sigma^2) - \Phi(k_{i_u};\bar{x};\sigma^2)) \cdot n - O_i)^2}{(\Phi(k_{i_o};\bar{x};\sigma^2) - \Phi(k_{i_u};\bar{x};\sigma^2)) \cdot n} \tag{6.4}$$

Da sich die Normalverteilung nicht elementar berechnen lässt, existiert diese in Tabellenform. Zur Implementierung in einem Algorithmus wird die beliebige Normalverteilung mit (6.5) in die Standardnormalverteilung überführt und mittels der Approximation nach Hastings (6.6) verwendet.

$$\Phi(x;\mu;\sigma^2) = \Phi(\frac{x-\mu}{\sigma};0;1) \tag{6.5}$$

$$\Phi(x;0;1) \approx \begin{cases} 1 - \frac{1}{\sqrt{2\pi}} \cdot e^{-x^2/2} \cdot (a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5) & \text{für } x \geq 0 \\ 1 - \Phi(-x;0;1) & \text{für } x < 0 \end{cases} \tag{6.6}$$

mit

$$t = \frac{1}{1+bx}, \qquad b = 0,2316419, \qquad a_1 = 0,31938153, \qquad a_2 = -0,356563782,$$
$$a_3 = 1,781477937, \quad a_4 = -1,821255978, \quad a_5 = 1,330274429.$$

Die Nullhypothese H_0 muss verworfen werden, falls

$$T > \chi^2_{p-1;1-\alpha}, \tag{6.7}$$

bzw. nicht verworfen werden, falls

$$T \leq \chi^2_{p-1-2;1-\alpha} \tag{6.8}$$

gilt. Auch die χ^2-Verteilung existiert nur in Tabellenform. Daher muss im Algorithmus eine Annäherung implementiert werden. Man bedient sich hierfür der Approximation nach Wilson und Hilferty:

$$\chi^2_{n;\gamma} \approx n \cdot \left[1 - \frac{2}{9n} + u_\gamma \cdot \sqrt{\frac{2}{9n}}\right]^3 \tag{6.9}$$

In (6.9) ist u_γ das Quantil der Standardnormalverteilung $N(0,1)$. Dieses wird über die Approximation von Hastings für Normalverteilungsquantile berechnet:

$$u_\gamma \approx t - \frac{a_0 + a_1 t + a_2 t^2}{1 + b_1 t + b_2 t^2 + b_3 t^3} \text{ mit } t = \sqrt{-2\ln(1-\gamma)} \quad \text{für } 0 < \gamma < 1 \tag{6.10}$$

mit

$$\begin{aligned} a_0 &= 2.515517, & a_1 &= 0.802853, & a_2 &= 0.010328, \\ b_1 &= 1.432788, & b_2 &= 0.189269, & b_3 &= 0.001308. \end{aligned}$$

Mit den bisher aufgestellten Berechnungen kann automatisch bestimmt werden, ob es sich bei Ausfall- und Reparaturdaten aus Produktionsdatenbanken um eine Normalverteilung handelt oder nicht. Statistisch verteilte Ereignisse in der Produktion weisen jedoch nicht nur normalverteiltes Verhalten auf, sondern man muss auch auf weitere Verteilungsfunktionen testen. Der implementierte Algorithmus, welcher sich innerhalb der mittlerer Schicht des entwickelten Frameworks aus Abschnitt 4.5.2 befindet, kann auf mehrere Verteilungsarten testen. Man kann hierfür die bereits gebildeten Klassenhäufigkeiten $O_1, ..., O_p$ nutzen. Lediglich bei der Berechnung von (6.3) muss eine andere Gleichung für den Erwartungswert E_i gewählt werden.

Für den Test auf eine Exponentialverteilung wird die Hypothese

$$H_0 : \text{die Grundgesamtheit ist } \exp(\lambda) \text{ verteilt}$$

gegen die Alternative

$$H_1 : \text{die Grundgesamtheit ist nicht } \exp(\lambda) \text{ verteilt}$$

aufgestellt. Dabei basiert der Test auch wieder auf einem bestimmten Signifikanzniveau α. Zur Berechnung der Prüfgröße setzt man in (6.3) den Erwartungswert der Exponentialverteilung (6.11)

$$F(x) = \begin{cases} \int_0^x f(y)dy = \int_0^x \lambda e^{-\lambda x} dy = 1 - e^{-\lambda x}, & \text{für } x \geq 0 \\ 0, & \text{für } x < 0 \end{cases} \tag{6.11}$$

ein und erhält die Prüfgröße (6.12):

$$T = \sum_{i=1}^{p} \frac{(\left[(1-e^{-\lambda k_{i_o}}) - (1-e^{-\lambda k_{i_u}})\right] \cdot n - O_i)^2}{\left[(1-e^{-\lambda k_{i_o}}) - (1-e^{-\lambda k_{i_u}})\right] \cdot n - O_i} \tag{6.12}$$

Bei der Exponentialverteilung wird nur der Parameter λ geschätzt. Dadurch erniedrigt sich der Freiheitsgrad bei der Berechnung des Schwellenwerts zur Annahme oder Ablehnung lediglich um 1 und es gelten folgende Aussagen:

Die Nullhypothese H_0 muss verworfen werden, falls

$$T > \chi^2_{p-1;1-\alpha}, \tag{6.13}$$

bzw. nicht verworfen werden, falls

$$T \leq \chi^2_{p-1;1-\alpha} \tag{6.14}$$

gilt.

Das Framework zur automatischen Generierung von simulationsrelevanten Daten kann nach dem dargestellten Schema auf weitere Verteilungen erweitert werden. Diese sollen jedoch nicht mehr mathematisch beschrieben werden. Hierzu gehören die Gleich- und die Poissonverteilung. Es soll jetzt noch näher auf ein Verfahren eingegangen werden, welches sich insbesondere für kleine Datenmengen oder Stichprobenumfänge eignet. Ein solches Verfahren ist unabdingbar, da sich bei bestimmten Ressourcen im betrieblichen Alltag nicht genügend Daten finden, um diese sinnvoll in Klassen einteilen zu können.

6.2.1.2 Kolmogoroff-Smirnov-Anpassungstest

Der Kolmogoroff-Smirnov-Anpassungstest ist ebenso wie der χ^2-Anpassungstest ein Prüfverfahren zum Test, ob eine Grundgesamtheit mit einer hypothetischen Verteilungsfunktion $F_0(x)$ übereinstimmt. Für kleine Stichprobenumfänge ist dieses Verfahren besser geeignet, als der χ^2-Test, da letzterer nur approximativ arbeitet (Klassenbildung) [Har98].

Die Hypothese

$$H_0 : F(x) = F_0(x) \qquad \text{für alle x}$$

gegen die Alternative

$$H_1 : F(x) \neq F_0(x) \qquad \text{für wenigstens einen Wert von x}$$

wird getestet mit der Prüfgröße

$$\sqrt{n}\, D_n, \text{ mit } D_n = \sup_x |F_0(x) - S_n(x)|, \tag{6.15}$$

wobei $S_n(x)$ die empirische Verteilungsfunktion der Beobachtungen $x_1, ..., x_n$ bezeichnet.

$$S_n(x) = \begin{cases} 0 & x < x_i \text{ für alle } i = 1, ..., n \\ \frac{k}{n}, \text{ falls} & x \geq x_i \text{ für genau k Beobachtungen } x_i \text{ aus } x_1, ..., x_n \\ 1 & x \geq x_i \text{ für alle } i = 1, ..., n \end{cases} \tag{6.16}$$

Die Größe D_n gibt folglich den größten vertikalen Abstand zwischen hypothetischer und empirischer Verteilungsfunktion an. Die Hypothese H_0 wird nun zum Niveau α verworfen, wenn gilt

$$\sqrt{n}\, D_n \geq d_{n;1-\alpha}$$

wobei die Quantile $d_{n;1-\alpha}$ wieder nur in empirischen Tabellen existieren. Ist ein vorliegendes n nicht in der Tabelle zu finden, so ist das Quantil zum nächstgrößeren n zu wählen. Bei der Normalverteilung werden μ und σ^2 durch $\bar{x}$ und s^2 aus der Stichprobe geschätzt, wodurch nicht die normalen Quantile aus der Tabelle verwendet werden können. Die Bestimmung der kritischen Werte selbst ist mit Schwierigkeiten verbunden. Lillifors hat 1964 aus diesem Grund Simulationsstudien durchgeführt und Tabellen der simulierten Quantile bereitgestellt [Lil67]. Umfangreichere Simulationsstudien von Stephens ergaben die von Pearson/Hartley in 1972 angegebenen kritischen Werte [PH72] (siehe Tab. 6.1).

n	5	8	10	20	30	>30
$l^{norm}_{n;0,90}$	0,72	0,74	0,76	0,79	0,80	0,81
$l^{norm}_{n;0,95}$	0,76	0,81	0,82	0,85	0,88	0,89
$l^{norm}_{n;0,99}$	0,91	0,94	1,03	1,03	1,03	1,04

Tab. 6.1: Kritische Werte $l^{norm}_{n;1-\alpha}$ zum Test auf nicht spezifizierte Normalverteilung

Ebenda finden sich auch die kritischen Werte für die Exponentialverteilung (siehe Tab. 6.1). Für die genannten Werte gibt es kein mathematisches Näherungsverfahren, wie es beim χ^2-Test implementiert ist. Daher sind die genannten Tabellen direkt in den Algorithmus implementiert. Auch der Kolmogoroff-Smirnov-Test funktioniert für andere Verteilungen, jedoch soll nicht weiter auf diese eingegangen werden.

n	5	8	10	20	30	>30
$l^{exp}_{n;0,90}$	0,91	0,93	0,94	0,96	0,97	0,98
$l^{exp}_{n;0,95}$	0,99	1,02	1,03	1,05	1,06	1,08
$l^{exp}_{n;0,99}$	1,15	1,20	1,21	1,24	1,26	1,28

Tab. 6.2: Kritische Werte $l^{exp}_{n;1-\alpha}$ zum Test auf nicht spezifizierte Exp.-verteilung

Bei der Auswertung von Ausfalldaten werden die beschriebenen Testverfahren jeweils zum Signifikanzniveau $\alpha = 0.99$, $\alpha = 0.95$ und $\alpha = 0.90$ ausgeführt. Es wird dann bestimmt, welche Verteilung am wahrscheinlichsten nicht abgelehnt werden kann. Diese wird in die XML-Darstellung übernommen. Führen alle Tests mit allen Signifikanzniveaus zu einer Ablehnung,

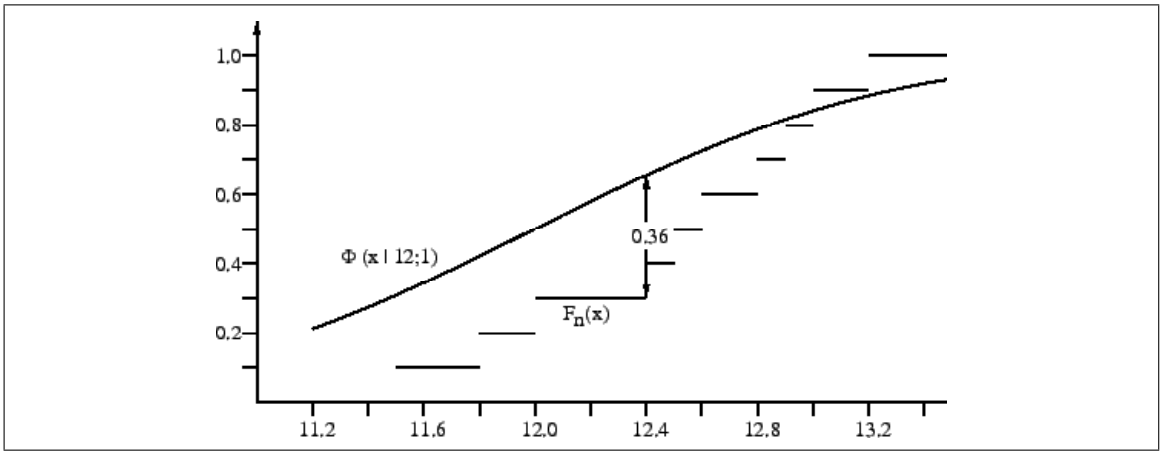


Abb. 6.12: Beispiel hypothetische und empirische Verteilungsfunktion (nach [Bol04])

so können entweder die diskreten Echtdaten übernommen werden, oder man kann sich für eine Standardverteilung mit gegebenen $\bar{x}$ und σ entscheiden. Übernimmt man die Echtdaten, so kann der Simulator diese als benutzerdefinierte Verteilung verwenden.

Die beschriebene und implementierte Systematik ist nicht nur für die Auswertung von Ausfall- und Reparaturdaten geeignet. Durch die Tatsache, dass die Middleware direkt XML-Daten lesen kann, ist es auch möglich, beliebige andere Zahlenreihen aus Produktionsdatenbanken auf statistische Verteilungen zu testen. Denkbar sind hier Bearbeitungszeiten aus Betriebsdaten-Erfassungssystemen (BDE), Rüstzeiten, Leitstandprotokolle und Werkerverfügbarkeiten. Der Algorithmus ist als Modul implementiert und kann daher schnell auf andere Daten angewendet werden. Er läuft als Server-Thread im Hintergrund und verarbeitet verschiedene Arten von übergebenen Daten. Die Ergebnisse werden in XML zurückgeschrieben.

6.2.1.3 Empirische Verteilungsfunktion

In bestimmten Fällen ist es nicht möglich mit den genannten Testverfahren eine Verteilung zu bestimmen. Es bleibt jedoch die Möglichkeit aus den Daten der Urliste eine empirische Verteilungsfunktion zu ermitteln [Sac97]. Dabei werden die Merkmalsausprägungen sortiert und die Häufigkeiten aufsummiert (vgl. Abb. 6.12). Simulationssysteme, wie Quest, Automod und Simflex/3D, können solche Funktionen für statische Verteilungen verwerten. Abhängig vom Simulationssystem werden diese Funktionen in Form von Listen als Textdatei (Quest, Simflex/3D) oder direkt im Simulationssystem (Automod) eingegeben.

6.2.2 Grenzen der Automatisierung

Eine Großzahl simulationsrelevanter Daten ist in verschiedenartigen Produktivsystemen vorhanden. Die entsprechenden Umsetzungsverfahren sind in Abschnitt 6.1 beschrieben. Darin enthalten sind Daten, wie Zeitbausteine, Layoutinformationen, Teilebedarfe und der logische Ablauf in der Fertigung. In anderen Bereichen stößt das Konzept jedoch an seine technischen Grenzen.

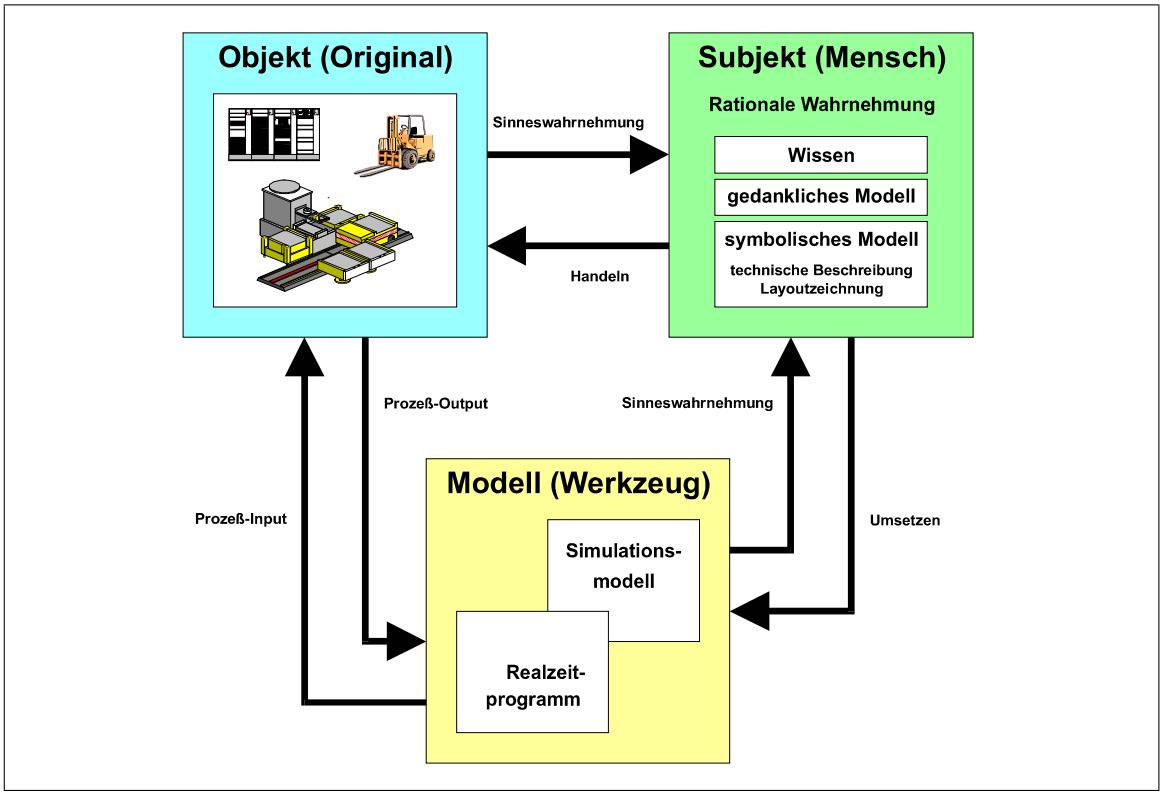


Abb. 6.13: Objekt-Subjekt-Modell-Relation [Rei88]

Bestimmte Umfänge an Daten, welche für die Simulation unverzichtbar sind, werden nicht in irgendwelchen Systemen gepflegt. Sie liegen in den subjektiven Entscheidungsbereichen von erfahrenen Meistern und Werkern und stellen einen Simulationsexperten vor die Tatsache, dass er Daten durch Beobachtung oder Interviews einholen muss. Hierzu zählen die Zuordnung von Werkern zu Maschinen, Priorisierungsstrategien bei der Maschinenauswahl, Notfallstrategien und viele weiteren subjektiven Faktoren in der realen Fabrik. Oftmals fällt es schon bei der manuellen Modellierung schwer, diese persönlichen Einflussfaktoren im Detail zu berücksichtigen. Andererseits ist es schließlich auch Aufgabe der Simulationsanwendung zu abstrahieren und neue Alternativen zu finden.

In Abb. 6.13 ist der Zusammenhang zwischen Objekt, Subjekt und Modell dargestellt. Das Bestreben bei der Anwendung von Simulationsstudien liegt stets darin, das reale Objekt aufgabenbezogen möglichst detail- und realitätstreu zu abstrahieren und abzubilden. In der Regel ist es jedoch nicht möglich, dieses Bestreben vollständig und objektiv umzusetzen. Die subjektiven Einflussfaktoren können zu unterschiedlichen Wahrnehmungen und Interpretationen führen, welche letztendlich jedem Simulationsmodell einen persönlichen Charakter verleihen. Jeder Mensch hat eine eigene Vorgehensweise bei der Erstellung eines Modells und dessen Umsetzung in einem Simulationssystem. Auch dieses bestimmt den Charakter eines Modells durch seine Philosophie mit. Jeder Simulator weist bestimmte Merkmale auf, durch welche er sich von anderen Produkten unterscheidet. Diese Softwaremerkmale bestimmen die Art und Weise, wie ein Objekt in ein Modell überführt wird.

Zusätzlich zu den subjektiven Einflussfaktoren gibt es noch weitere Bestandteile, welche eine automatische Modellgenerierung erschweren, oder welche sich anhand der vorhandenen Daten nicht realisieren lassen. Eine einfach erscheinende Aufgabe scheint die Zuordnung von Werkern zu bestimmten Maschinen. Angenommen die Dokumentationssysteme liefern über das im Rahmen dieser Arbeit entwickelte Framework Daten zu fünf Maschinen $M_1, ..., M_5$. Wir erhalten nun zu jeder Maschine nur jeweils einen Prozess $P_1, ..., P5$ mit den entsprechenden Mensch- und Maschinenzeiten, wie sie in Abschnitt 2.3 definiert sind. Zu den jeweiligen Prozessen erhalten wir noch den MS-Faktor, welcher festlegt, wie viele Prozesse dabei gleichzeitig von einem Werker bedient werden können. Weiter angenommen, die Prozesse P_1, P_2 und P_3 haben MS $= 3$ und P_4, P_5 haben MS $= 2$, so könnte man noch analytisch bestimmen, dass man zwei Werker auf die fünf Maschinen verteilen müsste. Jetzt gibt es jedoch auf jeder reellen Maschine weit mehr als einen Prozess und dabei noch je Maschine unterschiedliche MS-Faktoren. Daher kommt man schnell in einen Bereich, in dem analytische Methoden nicht mehr angewendet werden können. Heuristiken wären noch denkbar, aber in Anbetracht der Tatsache, dass es noch etliche Kombinationsmöglichkeiten der Prozesse gibt, und außerdem noch die technischen Qualifikationen der Menschen eine Rolle spielen, scheitern auch diese. Die Lösung kann erst während des dynamischen Simulationslaufs getroffen werden, wenn durch den Ablauf feststeht, welche Prozesse gerade auf den Maschinen laufen.

Ein Simulationssystem nimmt auf solche Tatsachen jedoch keine Rücksicht. Es muss mit programmierbaren Verfahren genau festgelegt werden, wie viele Werker im System sind, welche Prozesse sie bedienen können und evtl. noch die Prioritätsregeln. Insbesondere der in Abb. 6.14 dargestellte Steuerungsanteil, welcher eine der drei Komponenten eines Fabrikmodells ausmacht, ist nicht ohne Weiteres aus Produktions- und Dokumentationssystemen ableitbar. Die Zielsetzung dieser Arbeit besteht in der Umsetzung eines Datenframeworks zur teilautomatisierten Simulationsmodellgenerierung. Hierzu gehören die Bereiche Anlagen und Produkte aus Abb. 6.14 und auch der Inhalt des Fabrikmodells mit Produktionsprogrammen, Prozessen und Betriebsdaten. Basierend auf diesen automatisch aufbaubaren Grundbausteinen kann der Simulationsexperte in kurzer Zeit ein funktionsfähiges Modell erzeugen. Der Schwerpunkt der Nachmodellierung liegt dabei auf den logischen Steuerungsanteilen und Abweichungen von der in Abschnitt 5.2.1 definierten Basisgruppen zur Erzeugung eines Produktionsverbunds.

6.3 Erzeugen von Simulationsmodellen aus XML-Daten

Nachdem die Zusammenstellung der simulationsrelevanten Daten abgeschlossen ist und der Webserver entsprechende Anfragen beantworten kann, soll jetzt der Ablauf zwischen Webserver und Simulationsanwendung näher beschrieben werden. In Abschnitt 4.4.3 sind die bisherigen Entwicklungen im Bereich der standardisierten Modellbeschreibung näher beschrieben. Aus verschiedenartigen Gründen gibt es bis heute kein einheitliches Austauschformat für Simulationsmodelle. Diese Entwicklungsrichtung liegt auch nicht im Fokus der Softwareentwickler, da hierdurch Simulationssysteme sehr unkompliziert austauschbar wären und somit der Markt für die unterschiedlichen Anbieter verschärft würde. Daher ist in der nahen Zukunft nicht mit einer solchen Umsetzung zu rechnen.

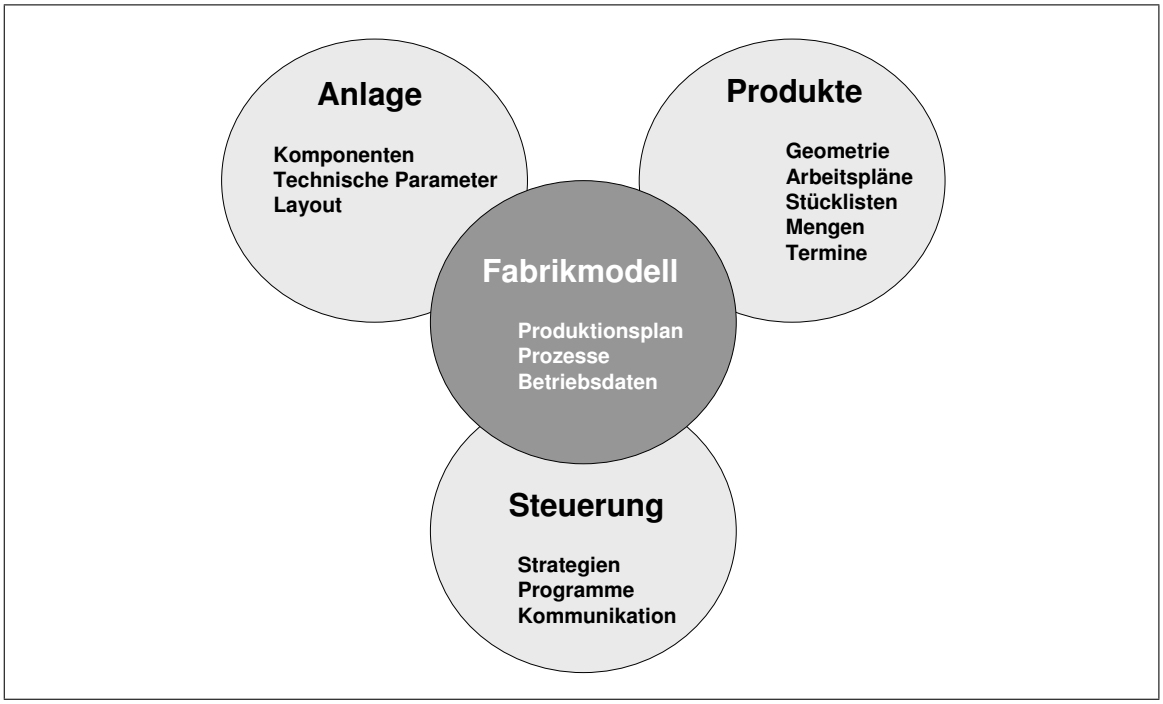


Abb. 6.14: Das Fabrikmodell und seine Komponenten [Rei89]

6.3.1 Standardisierte Darstellung von Modellen

Die Datenversorgung aus dem in dieser Arbeit entwickelten Konzept erfolgt in einer strukturierten Form in XML. In Abschnitt 4.4.2 ist in Abb. 4.10 am Beispiel einer Maschine die Struktur dargestellt. Abb. 6.6 zeigt diese Struktur befüllt mit Daten aus den Produktivsystemen. Eine Maschine ist dabei untergliedert in Daten zu ihrem Standort im Layout der Fabrik, in eine MTBF und eine MTTR, ihre Identifikationsnummer und den Typ von Beladeeinrichtung. Diese Strukturierung ist beliebig erweiterbar, wenn es gilt zusätzlich Werkzeuge, Betriebsmittel, etc. in die Datenstruktur aufzunehmen.

Ebenso, wie für Maschinen, gibt es weitere Strukturelemente für Teile und für Arbeitsvorgänge. Unterhalb der Teile befinden sich Elemente für die Jahresstückzahl, die zugehörigen Transportmittel mit den jeweiligen Fassungsvermögen und Losgrößen. Die Zuordnung der Teile zu entsprechenden Maschinen erfolgt innerhalb des Strukturelements Arbeitsvorgänge. Darunter befinden sich Bearbeitungs-, Belade-, Prüfzeiten und weitere relevante Daten (vgl. Abb. 6.15). Auch diese fließen wieder in das Gesamtschema aus Abb. 4.10 ein. Weiterhin gibt es noch bereichsübergreifende Daten, die zu berücksichtigen sind. Hierzu gehören u.a. die unterschiedlichen Schichtmodelle. In der Realität kann es durchaus vorkommen, dass eine Fertigung im 3-Schichtbetrieb arbeitet, während ihr Kunde, z.B. die Montage, nur im 2-Schichtmodell produziert. Zwischen den Fertigungsbereichen und der Montage gibt es Pufferplätze, auch Supermärkte genannt. Diese dienen der Materialversorgung der nachfolgenden Bereiche. Für die Modellgenerierung ist hierbei die Kapazität von großer Bedeutung. Mit diesen Strukturelementen innerhalb XML lässt sich ein gutes Grundmodell für ein Simulationsexperiment in der Teilefertigung erzeugen.

```
<?xml version="1.0" encoding="UTF-8"?>
<AVO>
    <Teile-Nr>A3833510065</Teile-Nr>
    <Inv-Nr>J00210061</Inv-Nr>
    <AVO-Nr>801</AVO-Nr>
    <Benennung>Ablängen und zentrieren</Benennung>
    <PPD_Versionsnummer>V00.001</PPD_Versionsnummer>
    <Maschgruppe>F019243423342</Maschgruppe>
    <Ts>0.0</Ts>
    <Tsb>0.0</Tsb>
    <Tsm>0.0</Tsm>
    <Tp>0.0</Tp>
    <Te>1.7401</Te>
    <Tng>2.4</Tng>
    <Tun>2.0</Tun>
    <Ps>40.0</Ps>
    <Pz parallel="true">4.71</Pz>
    <Laderkap>10</Laderkap>
    <Werker>0.25</Werker>
</AVO>
```

Abb. 6.15: Exemplarische Darstellung eines Arbeitsvorgangs in XML

Der reale Zustand in der Fabrik ist entsprechend der Daten aus den Produktions- und Dokumentationssystemen in einer für die Simulation geeigneten Struktur vorhanden. Der Simulationsexperte begnügt sich jedoch nicht damit, einen von der Planung und Produktion generierten Ist-Zustand zu simulieren. Dieser Zustand ist nur der Ausgangspunkt für bestimmte Anpassungen, welche eine Simulationsstudie erforderlich machen, um entsprechende Alternativen zu vergleichen. Daher muss eine Möglichkeit geschaffen werden, die automatisch zur Verfügung gestellten Daten nachzubearbeiten, bevor diese in einem Simulationssystem in ein Modell überführt werden.

6.3.2 Grafische Nachbearbeitung der generierten Daten

Es gibt mehrere Verfahren strukturierte XML-Dateien programmiertechnisch einzulesen und weiterzubearbeiten. Man unterscheidet Parser, welche die Daten sequentiell analysieren und durcharbeiten, diese jedoch nicht im Speicher halten, und Parser, welche die Daten in strukturierter Form im Speicher ablegen. Die bekannteste Implementierung ist der SAX-Parser[6]. Dieser liest sequentiell XML-Dateien und ruft für definierte Ereignisse (z.B. Startmarke, Endmarke, etc.) selbst definierbare Callback-Funktionen auf. Diese Systematik bietet sich insbesondere bei der Umwandlung und Verarbeitung von XML-Daten an. SAX ist zustandslos und erlaubt keinen freien Zugriff auf die Inhalte einer XML-Datei. Ganz im Gegensatz hierzu steht das DOM-API[7]. Es besteht aus Knoten, die Dokumente, Elemente und Attribute enthalten und innerhalb einer Baumstruktur durch Zeiger miteinander verbunden sind.

[6] Simple API for XML processing
[7] Document Object Model

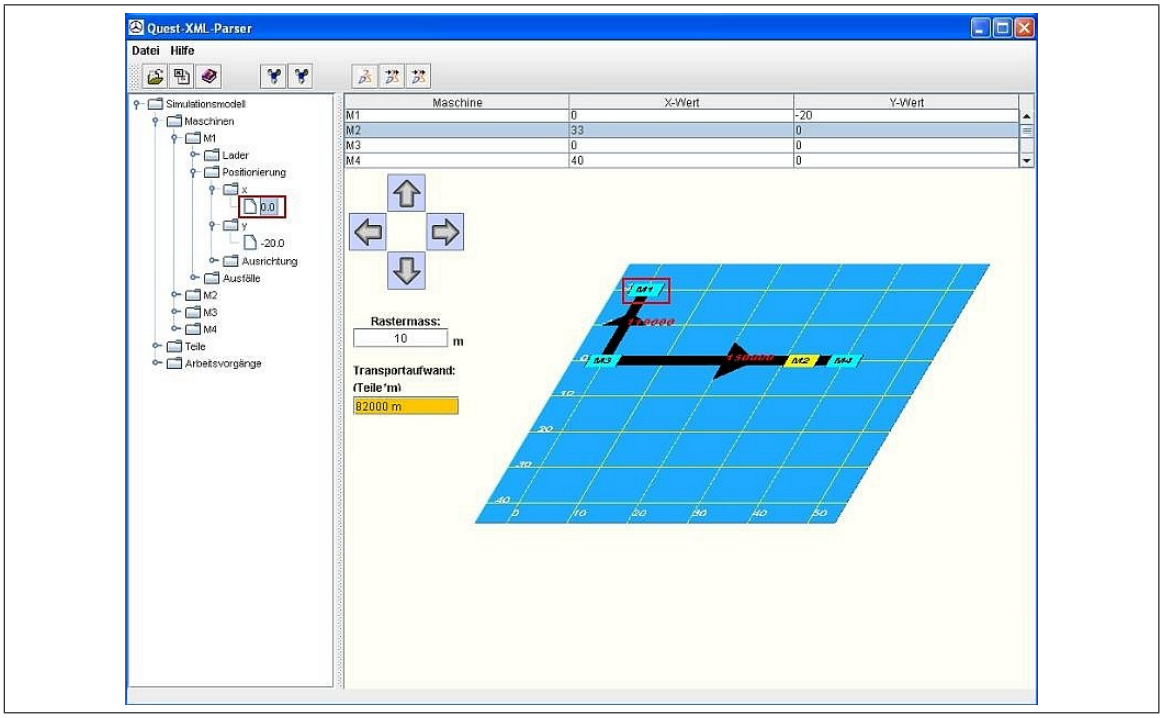


Abb. 6.16: Grafische Oberfläche zur XML-Bearbeitung, basierend auf JAXB

Für die Nachbearbeitung der Daten müssen diese im Speicher gehalten werden, um einen schnellen und wahlfreien Zugriff zu gestatten. Hierfür wurde das JAXB-API[8] gewählt, welches die Daten ähnlich wie DOM im Speicher hält [Wik07b]. In Erweiterung zu DOM bindet JAXB jedoch die XML-Elemente an zugehörige Java-Klassen. Dadurch wird es möglich, auch Methoden auf den Objekten zu erstellen und diverse Modellierungsfunktionalitäten zu implementieren. Eine dieser Möglichkeiten ist die Darstellung des Materialflusses als Sankey-Diagramm. Dabei werden die Materialströme zwischen Maschinen grafisch durch Pfeile angezeigt, welche in der Dicke entsprechend der Materialflussintensität skaliert werden (vgl. Abb. 6.16 und Abb. 6.17). Das Abbilden von XML-Daten auf Java-Objekte bietet sich in diesem Fall besonders an, da die Inhalte bei dem entwickelten Simulationsdatenframework komplex sind und auch von Zeit zu Zeit erweitert werden sollen. Es entfällt das aufwändige Ändern der Programmteile zur Nachbearbeitung und Umwandlung der generierten Inhalte.

Die in Abb. 6.16 dargestellte, im Rahmen dieser Arbeit programmierte, grafische Oberfläche zur Nachbearbeitung der Modelldaten, stellt in der linken Spalte den Strukturbaum der XML-Daten editierbar dar. Die Daten werden von dem entwickelten Datenframework vom Webserver bezogen. Durch Auswahl der Elemente innerhalb dieses kann man zu den gewünschten Daten navigieren und diese ändern, löschen oder auch neue hinzufügen (vgl. Abb. 6.17. Innerhalb dieser Oberfläche ist es möglich, bestehende Daten eines Fertigungsbereichs mit denen einer neu geplanten Maschine oder Anlage zu ergänzen. Außerdem kann dieses Werkzeug verwendet werden, um einen schnellen Überblick über Materialflussströme in den entsprechenden Bereichen und die zusammengeführten Daten der bestehenden Fertigung zu erhalten. Weiterhin wird hier

[8] Java Architecture for XML Binding

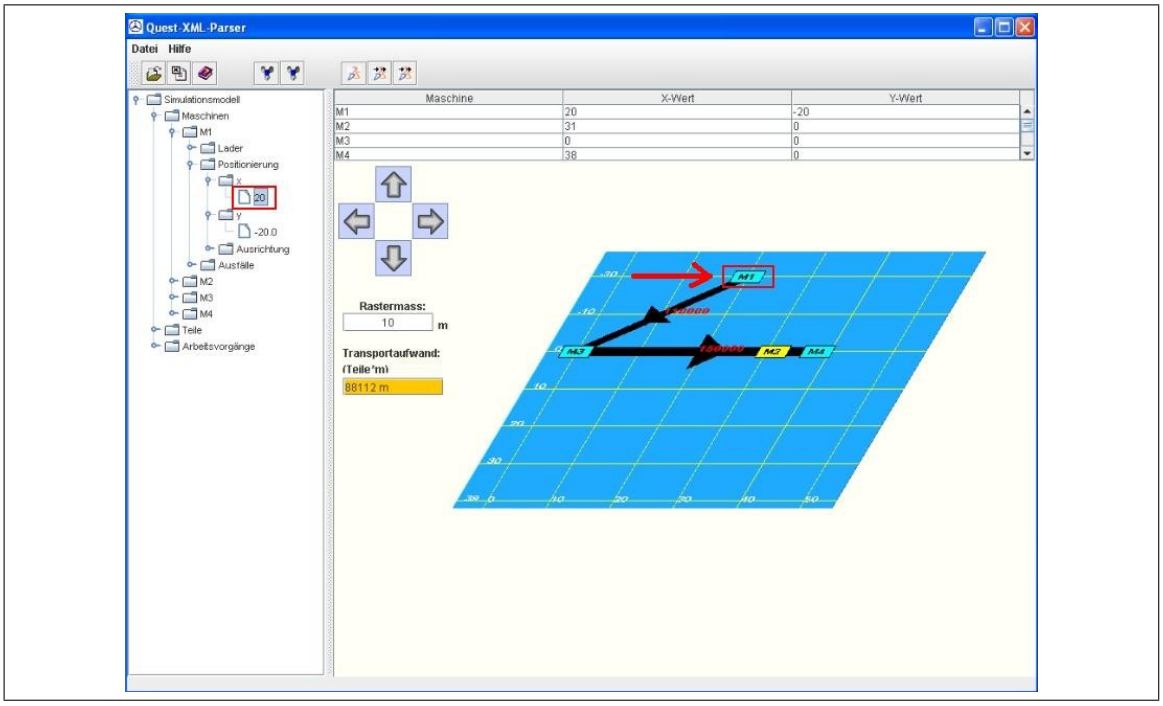


Abb. 6.17: Grafische Oberfläche zur XML-Bearbeitung nach Datenänderung

eine einfache Layoutplanung zur Verfügung gestellt. Die im rechten Rahmen der Anwendung dargestellten Ressourcen können entweder durch ziehen mit der Maus verschoben werden, oder auch in der Baumstruktur mit exakten Werten eingegeben werden. Ein Verschieben der Ressourcen wirkt sich dabei sofort auf die XML-Daten im Speicher aus. Bestimmte Planzustände können als XML-Datei in gleicher Form, wie sie vom Simulationsdatenframework erzeugt werden, gespeichert werden.

Die Materialflusssimulation ist ein Werkzeug zur Bewertung verschiedener Planungsalternativen und entfaltet daher ihren vollen Nutzen erst durch Wiederholungs- und Anpassungsschleifen. Es wird ein bestimmter Zustand simuliert und anhand der Ergebnisdaten werden Rückschlüsse auf die zu verändernden Parameter gezogen. Daher wird im Rahmen dieser Arbeit nicht direkt aus den Produktionsdaten ein Simulationsmodell erzeugt, sondern die genannte grafische Nachbearbeitungsebene implementiert. In Abb. 6.18 ist das gesamte Verfahren zur teilautomatisierten Generierung von Simulationsmodellen aus den industriellen Datenbanksystemen schematisch dargestellt. Auch die Applikation zur Nachbearbeitung der Daten basiert dabei auf Java und kann mittels Intranet oder Internet zur Verfügung gestellt werden. Aufgrund der Entscheidung, dass es zwischen Server und Simulationssystemen eine Nachbearbeitungsebene geben muss, folgt analog, dass auch die Modellerzeugung auf dieser Stufe erfolgen muss. Es wären auch andere Ansätze denkbar. So könnte der Simulator auf einem entfernten Rechner im Netzwerk laufen und die Parameter würden diesem über Netzwerk zur Verfügung gestellt. Es ist jedoch sinnvoll die Modellerzeugung und den Kreislauf der Alternativenvergleiche auf ein und demselben Rechner zu halten, da diese Lösung den geringsten Datentransfer erzeugt.

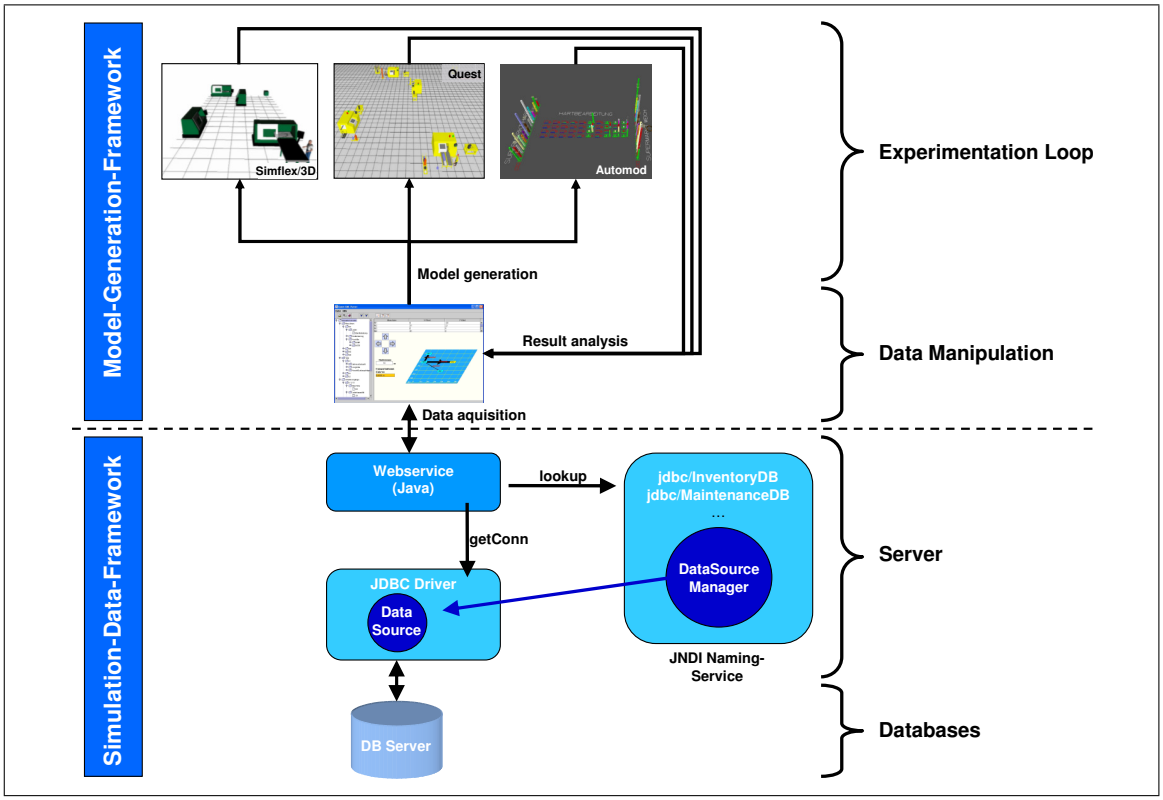


Abb. 6.18: Zusammenhang Simulationsdatenframework und Modellgenerierung

6.3.3 Modellgenerierung in verschiedenen Simulationssystemen

Viele der kommerziellen und nichtkommerziellen Simulationssysteme für Fabrikplanung basieren auf der in Abschnitt 2.1.1 beschriebenen, diskreten ereignisgesteuerten Philosophie. Somit sind sie bzgl. Ablauf und Zeitsteuerung sehr ähnlich, jedoch muss man in der Praxis enorme Unterschiede bei der Vorgehensweise zur Modellierung und der Auslegung der Anwendungsschwerpunkte feststellen. Diese Unterschiede lassen sich unter anderem über die Historie mancher Simulatoren erklären. Währen Quest ursprünglich ein CAD-System war, welchem ein Simulationskernel hinzugefügt wurde, so ist emPlant ein Simulator, welcher von einer alphanumerischen Version ausgebaut und mit Verknüpfungen zu CAD-Objekten erweitert wurde.

So unterschiedlich die Simulatoren in sich sind, so verschieden sind auch die Möglichkeiten, automatisiert in diesen Simulationsmodelle zu erzeugen. Auch hier können wieder die Vorzüge der Simulationsdatenhaltung in XML genutzt werden. Die Parser für diese Daten sind programmiertechnische Grundbausteine (API), die XML-Daten sowohl sequentiell als auch wahlweise analysieren und beim Auftreten bestimmter Ereignisse selbst definierbare Methoden aufrufen. Anschaulich bedeutet das, wenn ein Parser auf das Element `Maschine` innerhalb der Simulationsdaten trifft, so ruft er eine vordefinierte Programmroutine auf und meldet dieser das Ereignis `StartElement` mit dem Attribut `Maschine`. Die Routine weiß damit, dass jetzt die Unterelemente für eine Ressource vom Typ Maschine kommen. Der Programmcode für die Reaktion auf die Ereignisse muss vom Parserentwickler selbst erstellt werden. Nachdem alle Unterele-

mente bearbeitet sind, meldet der Parser das Ereignis `EndElement` wiederum mit dem Attribut `Maschine`.

Zwischen den Hauptelementen verläuft das Verfahren ebenso. Es werden dann Ereignisse, wie `StartElement` mit Attribut `Positionierung` und ebenso für Ladertyp, MTTR, MTBF, etc. erzeugt. Diese Technik macht die Umwandlung von XML-Daten sehr flexibel. In der Praxis werden XML-Parser für die verschiedensten Anwendungen eingesetzt. So können standardisierte PDF-Dokumente aus dynamischen Daten erzeugt werden, aber auch komplexe, verteilte Datenaustauschsysteme, wie z.B. ebXML[9] als Nachfolger von EDI[10] und der neue Standard für Wetterdaten OMF[11] sollen hier genannt werden. In den folgenden Abschnitten sollen die Implementierungstechniken für die Simulationssysteme Quest, Simflex/3D und Automod beschrieben werden.

6.3.3.1 Parserimplementierung für Quest

Quest legt Modelldaten, bedingt durch die Tatsache, dass es auf einem CAD-System basiert, in sehr vielschichtiger Architektur ab. Es gibt eine Hauptdatei, in der die Modellstruktur gespeichert wird und Referenzen auf Logiken, CAD-Objekte, Kinematiken, etc.. Die Modelldateien beinhalten sehr viele unstrukturierte, alphanumerische Parameter, welche hauptsächlich für die grafische Darstellung verwendet werden. Ohne Spezifikation des Herstellers ist es nicht möglich, die Bedeutung dieser Werte zuzuordnen. Daher ist es auch nicht möglich Daten aus dem XML-Format direkt in ein für Quest lesbares Format zu überführen.

Der Simulator stellt jedoch zur Automatisierung bestimmter Vorgänge eine Art Makrosprache zur Verfügung, mit der jegliche Systemfunktionen ausgeführt werden können. Diese sogenannte „Batch-Control-Language“ (BCL) kann entweder Befehle aus einer Datei oder über Netzwerk per Socket-Schnittstelle empfangen. In Abschnitt 5.1 sind die Basiselemente für die automatisierte Modellgenerierung beschrieben. Diese werden ohne Parameter, wie Bearbeitungszeiten, Werkerzuordnungen, etc. in Quest erstellt und als Submodelle gespeichert. Mittels der BCL können diese Submodelle beliebig oft in ein zu generierendes Modell geladen und dann entsprechend der Positionierungsdaten aus XML in diesem verschoben werden. Anschließend werden die abstrakten Basiselemente mit den entsprechenden Parametern aus dem Simulationsdatenframework und der grafischen Oberfläche zur Nachbearbeitung versorgt.

Zur Umsetzung der übergreifenden Steuerungslogik aus Abschnitt 5.1.4 wird zusätzlich zu den Basiselementen ein Submodell für die Steuerung erzeugt. Dieser Schritt ist notwendig, um eine segmentierte ConWIP-Steuerung, Quellen- und Senkenlogik, sowie weitere logische Abläufe im Gesamtmodell zu implementieren. Zu Beginn des sequentiellen Lesevorgangs eines Parsers wird das Ereignis `StartElement` des Wurzelelements `Modell` aufgerufen. In der zugehörigen Programmroutine bedeutet dies, dass eine Socket-Verbindung zum Simulator aufgebaut und das Submodell für die Steuerungslogik, welches nur aus Quelle, Senke und den im Rahmen dieser Arbeit programmierten Simulationslogik besteht, geladen werden muss. Beim Eintreten

[9] Electronic Business XML – Einheitlicher Datenaustausch zwischen Unternehmen
[10] Electronic Data Interchange
[11] Weather Observation Definition Format

des Ereignisses `Maschine` wird entsprechend der Ladertechnik das Submodell für eine hand-, band- oder automatikbeladenen Maschine geladen. Dabei übermittelt der Parser per Socket-Schnittstelle den BCL-Befehl zum Laden des passenden Submodells. Diese Schritte werden analog auch für die Teilegenerierung, Arbeitsvorgänge, Schichtmodelle, etc. wiederholt. Sobald das Ereignis `EndElement` des Wurzelelements `Modell` eintritt, kann über BCL ein Simulationslauf gestartet werden und nach vollständigem Durchlauf auch auf statistische Daten zurückgegriffen werden. Im folgenden Beispielcode ist in der BCL-Syntax das Öffnen einer Socketverbindung zum Parser und das anschließende Generieren eines Maschinenelements in Teilen dargestellt. Es wird zuerst das passende Submodell (siehe Abschnitt 5.1.1) in das Gesamtmodell integriert. Danach wird dieses im Layout an die in XML definierte Position verschoben und gedreht. Anschließend wird die Ausfallverteilung erzeugt und die Prozesse werden definiert. Im folgenden Beispiel ist ein Auszug der über die Socket-Schnittstelle übermittelten Befehle in BCL dargestellt:

```
OPEN CLIENT 'localhost:5001' FOR UPDATE AS 30
IF (lader=='Hand') THEN BCL("MERGE MODEL '/hand.mdl') ENDIF
IF (lader=='Band') THEN BCL("MERGE MODEL '/band.mdl'") ENDIF
IF (lader=='Auto') THEN BCL("MERGE MODEL '/auto.mdl'") ENDIF
BCL("TRANSLATE GROUP TO xpos,ypos")
BCL("ROTATE GROUP ABOUT Z_AXIS BY angle")
BCL("ADD FAILURE distribution TO machname")
BCL("SET machname NUMBER OF PROCESSES TO num_teile")
BCL("SET machname CAPACITY TO capacity")
```

In Abb. 6.19 ist der schematische Aufbau einerseits für den Modellaufbau in Quest, und andererseits in Simflex/3D dargestellt. Technisch bedeutet die Umsetzung des Modellgenerierungskonzepts, dass für jeden Simulator ein entsprechender Parser entwickelt werden muss. Der Vorteil liegt in der gleichen Datengrundlage aus dem Simulationsdatenframework. Diese Technik schafft herstellerunabhängige Flexibilität und kann bei Veränderungen ohne großen Aufwand für den entsprechenden Simulator angepasst werden, ohne dass komplexe Programmteile durchgearbeitet werden müssen.

6.3.3.2 Parserimplementierung für Simflex/3D

An der Universität Kassel, im Fachgebiet für Produktionssysteme, wurde der Fabriksimulator Simflex/3D entwickelt. Dieser arbeitet modellierungstechnisch mit verschiedenen Modelldateien im Textformat. Die jeweiligen Dateien beinhalten Daten zu Ressourcen, Arbeitsplänen, etc.. Beim Öffnen eines Simulationsmodells liest der Simulator die Modelldateien und erzeugt daraus ein dreidimensionales Simulationsmodell, basieren auf bestimmten Standardbausteinen, wie Quelle, Drehpuffer oder Stauförderer.

An der Universität Kassel hat Herr Nemrude Verzano die Implementierung des Parsers für die in dieser Arbeit entwickelte Simulationsdatenstruktur auf Simflex/3D umgesetzt. Aufgrund der

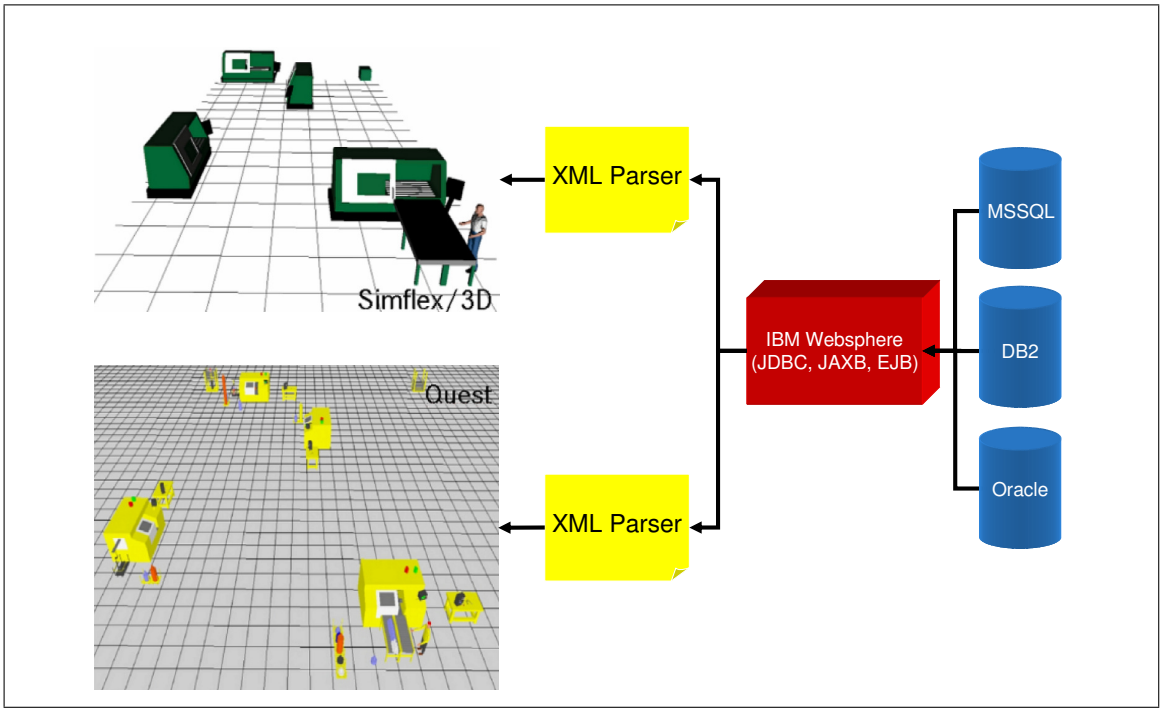


Abb. 6.19: Modellaufbau in Quest und Simflex/3D

Tatsache, dass dieser Simulator an der Hochschule entwickelt wurde, liegt hierfür der Quellcode vor. Daher konnte die Implementierung des Parsers direkt im Programm erfolgen. Es kann entweder direkt im Simulator eine von dem Simulationsdatenframework generierte XML-Datei geöffnet werden, oder man stellt die XML-Datei in ein Verzeichnis und übergibt dem Simulator die Referenz zu dieser Datei beim Aufruf.

Nachdem der Parser direkt in der Applikation implementiert ist, wird dieser automatisch beim Aufruf einer XML-Datei gestartet. Simflex/3D basiert jedoch auf anderen Standardbausteinen, als Quest. Daher müssen Merkmale, wie Bandlader auf die Simflex/3D Element Stauförderer, umgesetzt werden. Die Beschreibung einer Maschinenressource, wie sie in Abb. 6.6 dargestellt ist, wird bei der Überführung mit zusätzlichen Elementen für diesen Simulator versorgt, wie sie in Abb. 6.20 dargestellt sind. Diese Versorgung wird im entsprechenden Parser in der Routine zur Behandlung von den Elementen vom Typ Maschine implementiert.

Mit dieser Methodik kann ein Modell gleicher Struktur und mit gleichen Parametern sowohl in Quest, als auch in Simflex/3D erzeugt werden. Verschiebt man in der grafischen Oberfläche zur Datennachbearbeitung eine Maschine, so wird diese nach dem Erzeugen auch in den beiden Simulationssystemen entsprechend verschoben modelliert.

6.3.3.3 Parserimplementierung für Automod

Das Simulationssystem Automod ist, im Vergleich zu den beiden bisher genannten, nicht in der Lage, Modelle dynamisch zu erzeugen. Der Grund hierfür liegt im Aufbau des Simulators. Automod erzeugt aus der Modellierungsoberfläche eine ausführbare Datei des Modells. Ist diese

```
<?xml version="1.0" encoding="UTF-8"?>
<Modell>
    <Quelle>
        <Inv-Nr>Q5</Inv-Nr>
        <Sfx-Bezeichnung>Quelle</Sfx-Bezeichnung>          %% Simflex/3D Erweiterung
        <Sfx-Bausteintyp>1</Sfx-Bausteintyp>               %% Simflex/3D Erweiterung
        <Sfx-Variante>40</Sfx-Variante>                    %% Simflex/3D Erweiterung
        <Positionierung>
            <Pos-X>20.0</Pos-X>
            <Pos-Y>0.0</Pos-Y>
            <Pos-Z>0.0</Pos-Z>
            <Ausrichtung>N</Ausrichtung>
        </Positionierung>
    </Quelle>
    <Transportbaustein>
        <Inv-Nr>T6</Inv-Nr>
        <Sfx-Bezeichnung>Drehpuffer</Sfx-Bezeichnung>      %% Simflex/3D Erweiterung
        <Sfx-Bausteintyp>8</Sfx-Bausteintyp>               %% Simflex/3D Erweiterung
        <Sfx-Variante>10</Sfx-Variante>                    %% Simflex/3D Erweiterung
        <Positionierung>
            <Pos-X>40.0</Pos-X>
            <Pos-Y>0.0</Pos-Y>
            <Pos-Z>0.0</Pos-Z>
            <Ausrichtung>N</Ausrichtung>
        </Positionierung>
    </Transportbaustein>
    <Transportbaustein>
        <Inv-Nr>T7</Inv-Nr>
        <Sfx-Bezeichnung>Staufoerderer</Sfx-Bezeichnung>  %% Simflex/3D Erweiterung
        <Sfx-Bausteintyp>3</Sfx-Bausteintyp>               %% Simflex/3D Erweiterung
        <Sfx-Variante>40</Sfx-Variante>                    %% Simflex/3D Erweiterung
        <Positionierung>
            <Pos-X>20.0</Pos-X>
            <Pos-Y>0.0</Pos-Y>
            <Pos-Z>0.0</Pos-Z>
            <Ausrichtung>N</Ausrichtung>
        </Positionierung>
    </Transportbaustein>
</Modell>
```

Abb. 6.20: Erweiterung des Frameworks um Simflex/3D Bausteine

jedoch erstmal kompiliert, so können keine strukturellen Veränderungen mehr vorgenommen werden. Der Vorzug dieser Technik liegt jedoch in der Geschwindigkeit der Experimentdurchführung. Die compilierten Anwendungen laufen schneller als vergleichbare in einer Simulatorumgebung, wie in Quest.

In Abschnitt 5.2.1 wurde bereits dargestellt, wie in Automod Modelle über zusätzliche Dateien parametrisiert werden können. Der in diesem Abschnitt genannte Matrixaufbau kann somit ohne Parameter für Zeiten, Losgrößen, etc. kompiliert werden und liest diese zur Laufzeit ein. Die Grenzen dabei liegen jedoch bei der Layoutgenerierung. Es ist nach der Modellerzeugung nicht mehr möglich, Ressourcen im Layout zu verschieben. Das wirkt sich insbesondere bei der Simulation von Arbeitskräften aus, da hier die korrekten Wegstrecken nicht mehr über das Layout abgebildet werden können.

Aus den XML-Dateien werden Textdateien erzeugt, aus denen Automod zur Laufzeit die Modellparameter auslesen kann. Ein generisches Modell, in dem alle Ressourcen fest definiert werden müssen und dann entsprechend ein Teilbereich aus jenen deaktiviert wird, stößt jedoch sehr schnell an seine Grenzen und ist nicht ideal für die Modellgenerierung. Für die Zukunft ist es auch denkbar die Dateien, die aus der Modellierungsoberfläche von Automod generiert werden, anhand der XML-Dateien in einem Parser umzuschreiben. Der Simulator bietet die Möglichkeit die Kompilierung automatisch zu starten und anschließend das Modell auszuführen. Ohne die Spezifikationen zum Aufbau der Modelldateien ist diese Umsetzung jedoch ähnlich schwierig, wie bei Quest.

Kapitel 7

Anwendung in der Praxis

Die vorliegende Arbeit ermöglicht die Integration annähernd jedes beliebigen industriellen DV-Systems in eine strukturierte Datendarstellungsform. Hierdurch werden Zugriffsmöglichkeiten auf Planungs-, Produktions- und Dokumentationssysteme geschaffen. Die eingesetzten Techniken bieten vielfältige Möglichkeiten, Daten zu verknüpfen und in eine aufgabenorientiertere Form zu überführen. Dabei wurde als Kunde stets die Fabriksimulation in den Vordergrund gestellt. Im folgenden Abschnitt sollen weitere mögliche Einsatzfelder angesprochen werden, die bisher noch nicht im Detail genannt wurden. Im Anschluss daran werden die technischen Grenzen eines solchen Frameworks aufgezeigt.

7.1 Nutzung im Planungsumfeld

Die Möglichkeit der teilautomatisierten Erzeugung von standardisierten Grundmodellen wurde im Rahmen dieser Arbeit beschrieben und die Umsetzung in der Praxis getestet. Der streng hierarchische Aufbau (Abschnitt 5.2.2) und die übergeordnete Implementierung der Logik im Simulationsmodell (Abschnitt 5.2.3) erlauben Modelle unterschiedlichen Detaillierungsgrades. In Kapitel 1 und Abb. 1.2 ist der konventionelle Ablauf von Simulationsstudien als weitgehend manueller und zeitaufwändiger Bestandteil der Planung beschrieben. Der Simulationsexperte tritt als Dienstleister gegenüber den Planungsteams auf, da die Materialflusssimulation aus Zeit- und Qualifikationsgründen nicht von jedem Mitglied der Planung genutzt werden kann.

Im Rahmen dieser Arbeit wurde eine Methode zur teilautomatisierten Modellgenerierung aus Planungssystemen entwickelt und umgesetzt. Diese erlaubt es einem Planer mit Grundkenntnissen in der Materialflusssimulation standardisierte Simulationsmodelle zu nutzen und die Ergebnisse aus deren Ausführung zu interpretieren. Grundkenntnisse in Simulation sind notwendig, weil die generierten Daten bezüglich Aussagekraft, Detaillierungsgrad und Plausibilität überprüft werden müssen, da man nicht davon ausgehen kann, dass in den Dokumentationssystemen die Realität exakt genug beschrieben ist. Dem Simulationsexperten ermöglicht die Methode, schnell Daten aus verschiedenen Datenbanksystemen zusammenzuführen, diese zu überprüfen und anschließend in ein Grundmodell zu überführen. Der Aufwand, welcher bisher durch die Datensammlung und Modellierung entstanden ist, wird von einem Client-/Serversystem übernommen. Dadurch wird es möglich den Ansatz des Simultaneous Engineering auch im Bereich der Anwendung von Materialflusssimulation umzusetzen. Die Mitglieder der Planungsteams können wie gewohnt ihren Planungsaufgaben nachgehen und die ihnen bekannten Systeme zur

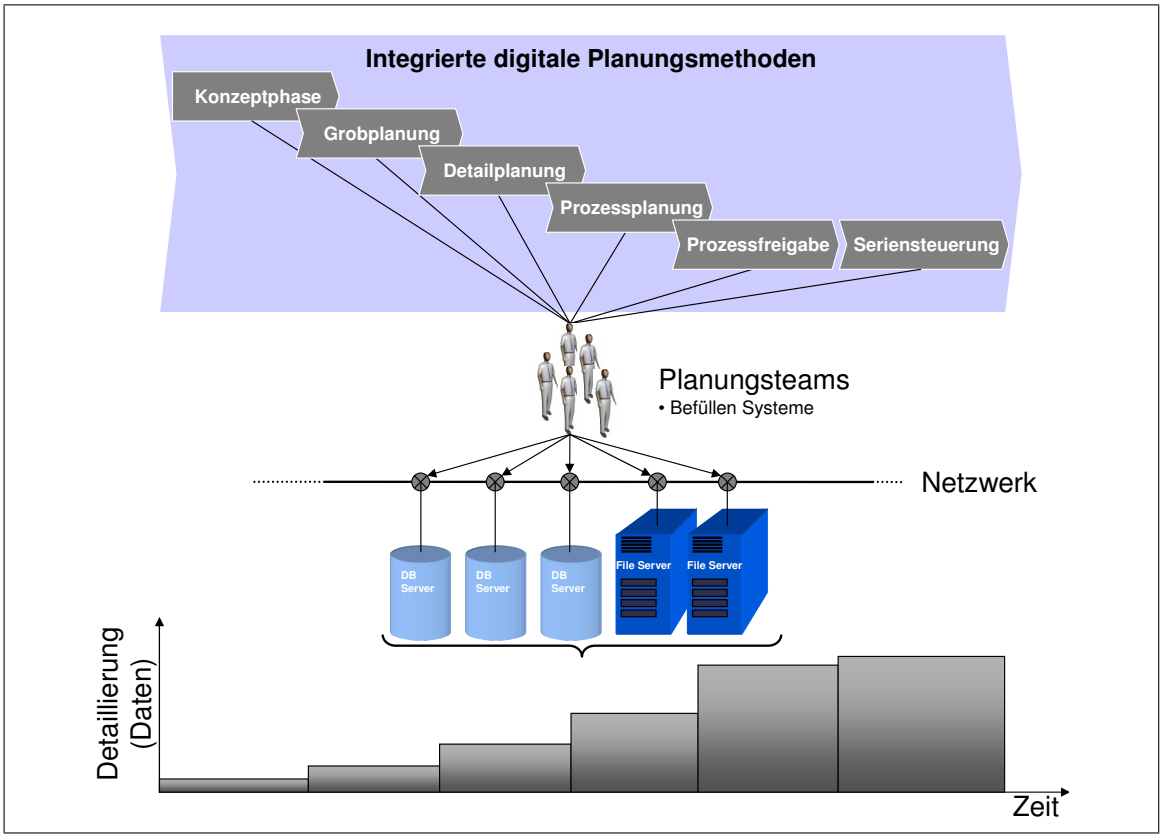


Abb. 7.1: Zusammenhang Planungsprozess und Datendetaillierung

Dokumentation und Datenablage nutzen. Der Simulationsexperte kann kollaborativ in den Planungsteams mitarbeiten, da er stets Zugriff auf die aktuellen Plandaten hat. Dabei steigt der Informationsgehalt in den Planungs- und Dokumentationssystemen stetig mit dem Fortschritt im Planungsprozess (vgl. Abb. 7.1). Die Kooperation zwischen Planungs- und Simulationsexperten wird vereinfacht und es bleibt mehr Zeit für die Bewertung von alternativen Szenarien.

Mit dem steigenden Informationsgehalt in den Planungssystemen können durch die Nutzung des entwickelten Simulationsdatenframeworks die Modelle mit wachsendem Detaillierungsgrad teilautomatisch angepasst werden. Während Simulationsstudien in der Vergangenheit nur einzelne Ausschnitte oder Zwischenstände des kontinuierlichen Planungsprozesses widergespiegelt haben, kann durch die Nutzung der vorgestellten Methode annähernd der gesamte Prozessablauf begleitet werden. Bei konsequenter Übertragung der Ergebnisse aus der Simulation in die Planung können Fehler frühzeitig erkannt und eliminiert werden. Die Verankerung der Materialflusssimulation als fester Bestandteil der Produktionsplanung wird wesentlich erleichtert (vgl. Abb. 7.2).

Der Erfolg dieser Integration hängt jedoch entscheidend von der Aussagekraft und dem Detaillierungsgrad der Daten in den Planungssystemen ab. Durch das teilautomatisierte Erzeugen von Modellen können sehr schnell Fehler und Inkonsistenzen in den Planungsdaten erkannt werden. Es lässt sich dann entweder kein Modell erstellen, da ein Laufzeitfehler im Programm auftritt oder aber das Modell ist aufgrund der Daten nicht lauffähig, weil logische Verbindun-

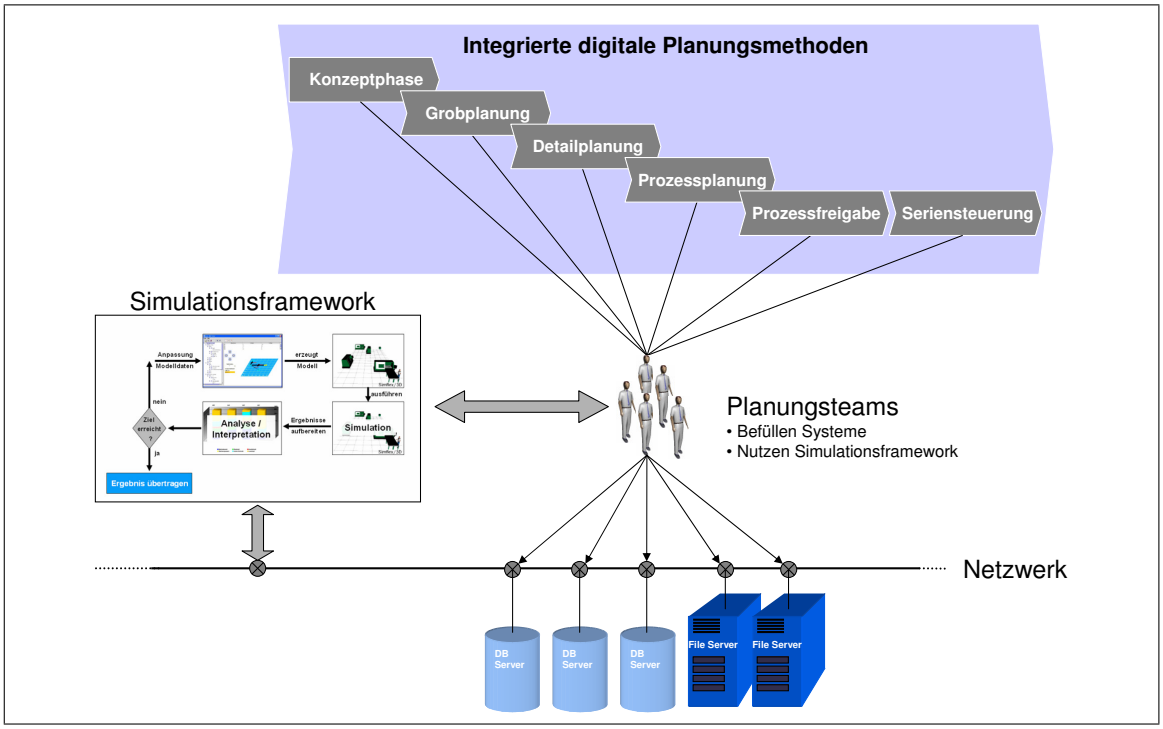


Abb. 7.2: Integration von Simulation in den Planungsprozess

gen nicht stimmen, Prozesszeiten fehlen, etc.. Planung ist ein kontinuierlicher Prozess, welcher im Laufe der Zeit verschiedene Schwerpunkte bzgl. Produkte und Bereiche beleuchtet. Es ist daher auch verständlich, dass nicht alle Planungsdaten zu jeder Zeit auf dem neuesten Stand sind. Im Umfeld der Fabrik trifft man auf ein Zusammenspiel vieler verschiedener Bereiche, welche jeder für sich eine bestimmte Autonomie und Eigenverantwortlichkeit besitzen. Unter diesen Umständen ist auch klar, dass eine Konsistenz in der Datenwelt, wie sie für die Simulation gefordert wird, kaum erreicht werden kann. Die Eingabe von Daten in Systeme erfolgen unter subjektiven Aspekten durch den Menschen. Somit erhält man bei der Nutzung der Daten aus diesen auch eine bestimmte Sichtweise darauf. Ein Logistikplaner plant beispielsweise eine Bereitstellungfläche für Teile unter dem Aspekt seiner Transportfahrzeuge, während der Materialflussplaner diese eher unter dem Gesichtpunkt seiner geplanten Losgrößen betrachtet. Den Montageplaner interessiert primär weder die Losgröße, noch das Transportfahrzeug, da er aus Platzgründen nur möglichst wenige Teile am Band haben möchte, welche jedoch stets verfügbar sein sollen. Der Simulationsexperte muss die Daten nach der Zusammenführung auf solche Abweichungen überprüfen, bevor ein Simulationsexperiment durchgeführt werden kann.

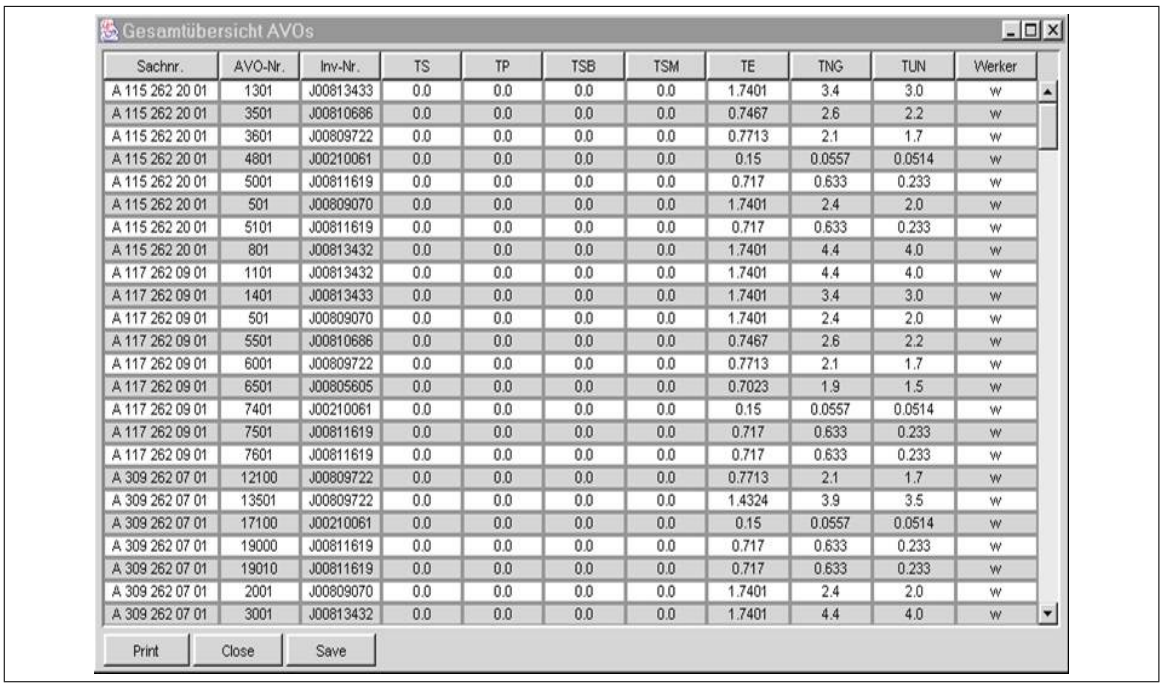
Ein einfaches Beispiel soll die Problematik der aus Sicht der Materialflusssimulation geforderten Datenqualität nochmals anschaulich machen. Angenommen es soll eine neue Anlage in einen bestehenden Bereich integriert werden. Mittels der Materialflusssimulation möchte man die Gesamtdurchlaufzeit, den Durchsatz und die Werkerzuordnung planen. Das Simulationsdatenframework liefert die ablaufrelevanten Daten des bestehenden Bereichs. Weiter angenommen der Produktionsmeister aus diesem Bereich hat vor geraumer Zeit aus Kapazitätsgründen ein Teil aus der dokumentierten Datenmenge an seinen Kollegen im Nachbarbereich abgegeben. Dort

ist das Teil im Dokumentationssystem jedoch nur als Ausweichfertigung oder Notfallstrategie deklariert. Dieser Kollege benötigt eine etwas längere Rüstzeit, da er einen anderen Maschinentyp einsetzt. Um den geplanten Rüstanteil nicht zu überschreiten, sammelt er die Teile, bis er ein größeres Los fahren kann. Anschließend kommen die Teile zurück in den zu betrachtenden Bereich.

Im Zusammenhang mit diesem Beispiel können in Bezug auf die Modellgenerierung zwei Fälle unterschieden werden: das Teil wurde von der Planung auf die momentane Maschine umgeplant und auch in der Dokumentation angepasst oder nicht. Beide Fälle führen zu Schwierigkeiten mit der entwickelten teilautomatisierten Modellierung. Angenommen die Systeme liefern die richtigen Daten, so entsteht durch das Verlassen des Betrachtungsbereichs eine Unterbrechung in dem Verbindungsnetzwerk für das Teilerouting aus Abb. 5.2 in Abschnitt 5.1.3. Strenggenommen müsste zur realitätstreuen Betrachtung des genannten Bereichs diese Fremdmaschine mit in das Modell integriert werden. Jedoch laufen über diese Ressource evtl. weitere Teile aus dem Nachbarbereich, welche dann auch berücksichtigt werden müssten. Die gesamte Fabrik ist eng vermascht, da heute eine flexible Produktion unabdingbar ist. Das hätte zur Folge, dass man sich bei der Integration der Maschine aus dem Nachbarbereich und Berücksichtigung derer Abhängigkeiten rekursiv durch die gesamte Fabrik arbeiten müsste. Diese Art der Rekursion wird im Simulationsdatenframework vermieden. Tritt der genannte Fall auf, so wird die nächstmögliche Maschine im Betrachtungsbereich gewählt und über die logische Verbindung verknüpft.

Ist die Maschine nicht umdokumentiert, so entsteht im Simulationsmodell der Fehler, dass die Maschine im Betrachtungsbereich überbelegt ist, obgleich dies in der Realität nicht der Fall ist. Das bedeutet gleichzeitig, dass in der Simulation nicht die in der Realität zu erwartenden Teilemengen auf die neue Anlage geliefert und somit der Durchsatz falsch ermittelt würde. Bei geringen Abweichungen, die nicht mit bloßem Auge erkennbar sind, könnte eine automatisierte Simulationsstudie schlimmstenfalls zu einer Fehlentscheidung führen. Der Planer kann Abweichungen in Dokumentationssystemen subjektiv erkennen bzw. weiß er aus Erfahrung, wie die Produktion gerade arbeitet. Im Gegensatz hierzu muss sich das Simulationsdatenframework auf die Datenqualität verlassen und diese algorithmisch interpretieren. Aufgrund der genannten Risiken wurde im Rahmen dieser Arbeit ein *teilautomatisierter* Aufbau gewählt. Die Methodik soll primär den Simulationsexperten unterstützen und bei überprüfter Datengrundlage auch vom Planer mit Grundkenntnissen in der Simulation verwendet werden.

Im Rahmen der planerischen Tätigkeiten wurde in Abschnitt 2.3 die Problematik der Informationsbeschaffung genannt. Planer müssen in verschiedenen Systemen ihre benötigten Daten suchen und von Hand zusammenstellen. Insbesondere bei der Durchführung von Planungsworkshops hängt die benötigte Zeit und das Gelingen stark von der Verfügbarkeit aller Daten eines zu betrachtenden Bereichs ab. Hierfür wurde in Erweiterung zu der vorliegenden Arbeit eine Oberfläche programmiert, welche online auf die Daten des Frameworks zugreifen kann (vgl. Abb. 7.3). Anstatt die Daten an einen Parser zu übergeben, werden diese über ein GUI dargestellt und können zur schnellen Datenbeschaffung aus mehreren Systemen genutzt werden. Ebenso können bestimmte Datenausprägungen im Intranet zur Verfügung gestellt werden. Durch den Einsatz von XML als Datenbasis wird es möglich, gezielt bestimmte Teilbereiche aus der standardisierten gesamten Datenmenge auszugeben, z.B. Maschinen mit zugehörigen Ausfällen für einen Maschinenplaner, oder Prozessgraphen für den Ablaufplaner (vgl. Abb. 7.4).



Sachnr.	AVO-Nr.	Inv-Nr.	TS	TP	TSB	TSM	TE	TNG	TUN	Werker
A 115 262 20 01	1301	J00813433	0.0	0.0	0.0	0.0	1.7401	3.4	3.0	w
A 115 262 20 01	3501	J00810686	0.0	0.0	0.0	0.0	0.7467	2.6	2.2	w
A 115 262 20 01	3601	J00809722	0.0	0.0	0.0	0.0	0.7713	2.1	1.7	w
A 115 262 20 01	4801	J00210061	0.0	0.0	0.0	0.0	0.15	0.0557	0.0514	w
A 115 262 20 01	5001	J00811619	0.0	0.0	0.0	0.0	0.717	0.633	0.233	w
A 115 262 20 01	501	J00809070	0.0	0.0	0.0	0.0	1.7401	2.4	2.0	w
A 115 262 20 01	5101	J00811619	0.0	0.0	0.0	0.0	0.717	0.633	0.233	w
A 115 262 20 01	801	J00813432	0.0	0.0	0.0	0.0	1.7401	4.4	4.0	w
A 117 262 09 01	1101	J00813432	0.0	0.0	0.0	0.0	1.7401	4.4	4.0	w
A 117 262 09 01	1401	J00813433	0.0	0.0	0.0	0.0	1.7401	3.4	3.0	w
A 117 262 09 01	501	J00809070	0.0	0.0	0.0	0.0	1.7401	2.4	2.0	w
A 117 262 09 01	5501	J00810686	0.0	0.0	0.0	0.0	0.7467	2.6	2.2	w
A 117 262 09 01	6001	J00809722	0.0	0.0	0.0	0.0	0.7713	2.1	1.7	w
A 117 262 09 01	6501	J00805605	0.0	0.0	0.0	0.0	0.7023	1.9	1.5	w
A 117 262 09 01	7401	J00210061	0.0	0.0	0.0	0.0	0.15	0.0557	0.0514	w
A 117 262 09 01	7501	J00811619	0.0	0.0	0.0	0.0	0.717	0.633	0.233	w
A 117 262 09 01	7601	J00811619	0.0	0.0	0.0	0.0	0.717	0.633	0.233	w
A 309 262 07 01	12100	J00809722	0.0	0.0	0.0	0.0	0.7713	2.1	1.7	w
A 309 262 07 01	13501	J00809722	0.0	0.0	0.0	0.0	1.4324	3.9	3.5	w
A 309 262 07 01	17100	J00210061	0.0	0.0	0.0	0.0	0.15	0.0557	0.0514	w
A 309 262 07 01	19000	J00811619	0.0	0.0	0.0	0.0	0.717	0.633	0.233	w
A 309 262 07 01	19010	J00811619	0.0	0.0	0.0	0.0	0.717	0.633	0.233	w
A 309 262 07 01	2001	J00809070	0.0	0.0	0.0	0.0	1.7401	2.4	2.0	w
A 309 262 07 01	3001	J00813432	0.0	0.0	0.0	0.0	1.7401	4.4	4.0	w

Abb. 7.3: GUI zur Darstellung der generierten Daten

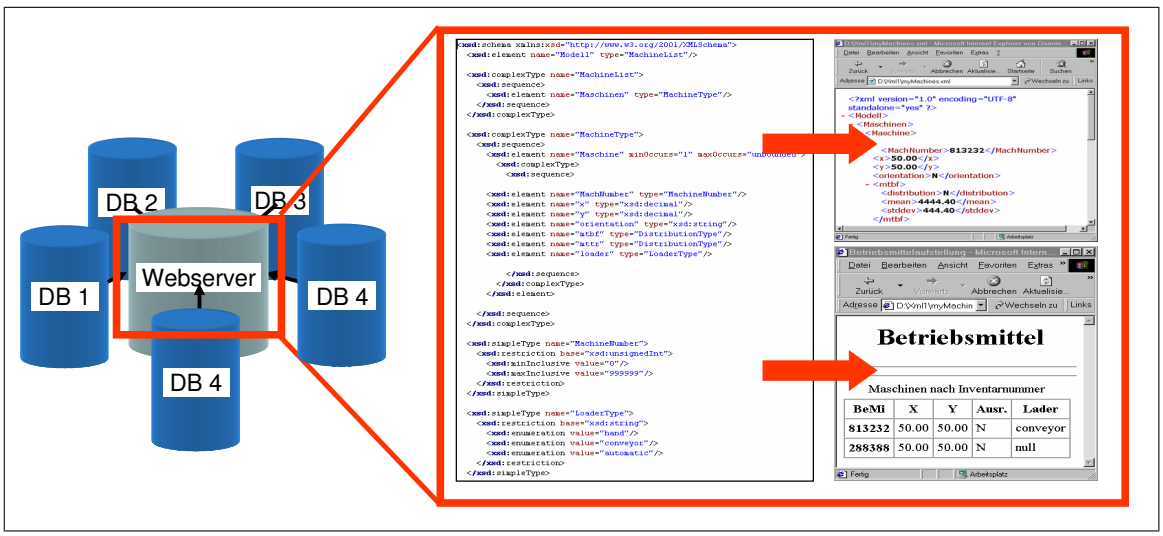


Abb. 7.4: Darstellung bestimmter Teilmengen aus den Gesamtdaten

7.2 Automatische Datenkonsistenzprüfung

Die Qualität großer Datenbestände in der Industrie hängt von einer Vielzahl von Faktoren ab. Dateneingabe und Datenbeschaffung bringen stets auch Fehler mit sich, die von einfacher und komplexer Art sein können. Unternehmen, die nicht extreme Vorkehrungen treffen, um diese Fehler zu vermeiden, weisen typische Fehlerraten um die 5% auf [Red98, Orr98]. Die logische Folgerung ist, dass Unternehmen permanent damit beschäftigt sind, Daten zu bereinigen und Inkonsistenzen zu beseitigen. Manuell ist dieser Tatsache jedoch nicht zu begegnen, da ein gewaltiger Personalaufwand benötigt würde. Schon seit Jahren geben Unternehmen jährlich große Beträge aus, um Fehler in den Datenbeständen zu erkennen und zu beseitigen [Red96].

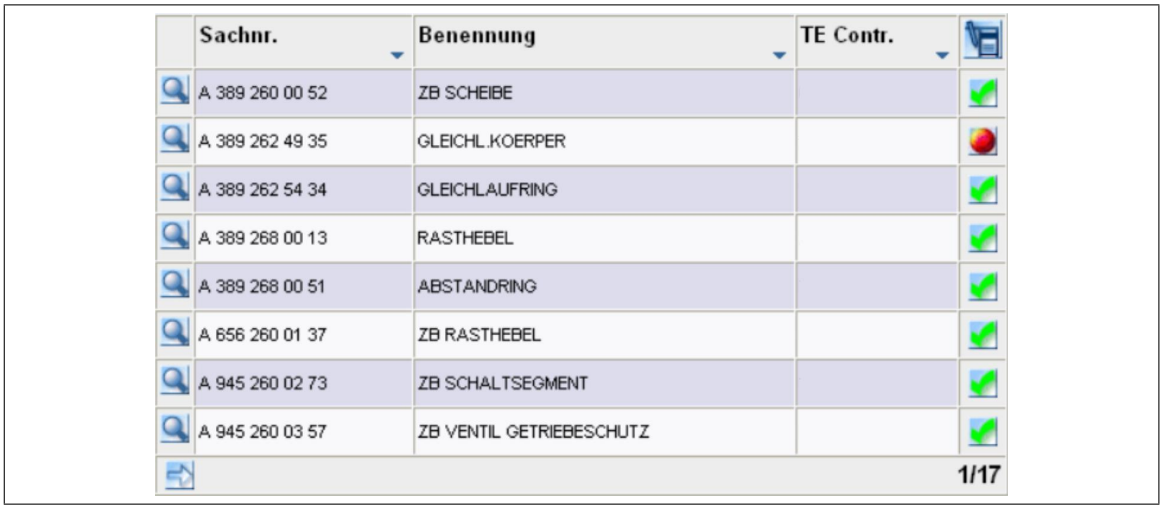
Das Simulationsdatenframework erlaubt das Zusammenführen von Daten aus verschiedenen Systemen. In Abschnitt 4.5.2.2 wurde die Problematik unterschiedlicher Darstellungen gleicher Inhalte erwähnt. Wenn sich Daten nur in ihrer Darstellung unterscheiden (z.B. mit/ohne Leerzeichen) so handelt es sich noch nicht um eine Inkonsistenz. Diese Art der Fehler werden innerhalb der Reformulation-Engine (Abschnitt 6.1.2.1) und der Execution-Engines (Abschnitt 6.1.2.2) durch reguläre Ausdrücke umgangen. Im Umfeld der Produktionsplanung ist wichtig, Daten trotz kontinuierlicher Veränderungen in der Fertigung konsistent zu halten.

Ein typisches Beispiel findet sich im Bereich der Fertigungspläne. Maschinen werden im Fertigungsplan entsprechenden Prozessen zugeordnet. Dabei können unterhalb eines Prozesses sowohl Maschinen für die Serien- als auch für die Ausweichfertigung definiert werden. Über die Jahre und bedingt durch den Lebenszyklus der Ressourcen werden neue Anlagen beschafft und alte verkauft oder verschrottet. Das hat zur Folge, dass innerhalb des Fertigungsplans Maschinen auftauchen können, welche bereits nicht mehr in der Fabrik sind. Das stellt einerseits eine Schwierigkeit für die Modellgenerierung dar und andererseits kann es dazu führen, dass mit dem Ausscheiden alter und der Neueinstellung junger Planer der Überblick verloren geht.

Zur Vermeidung dieser Fehler wird mittlerweile das im Rahmen dieser Arbeit entwickelte Verfahren zur Datenzusammenführung verwendet. Auf Intranetbasis können Unstimmigkeiten zwischen verschiedenen Datenbanken visualisiert werden und als Grundlage für die Datenbereinigung genutzt werden (vgl. Abb. 7.5). Man muss nicht mehr verschiedene Listen vergleichen und sogar vor Ort eine Inventur durchführen, um herauszufinden, an welchen Stellen man Korrekturen vornehmen muss.

7.3 Einsatz als Datenquelle für andere Systeme

Es wurde bereits erwähnt, dass der Interaktion zwischen verschiedenen Softwaresystemen innerhalb von Unternehmen große Bedeutung zugemessen wird. Einerseits ist es technisch nicht möglich, mit einer einzigen Softwarelösung auszukommen, die vom Leitstand bis zur Lohnabrechnung alles erledigen kann. Andererseits muss man die Systeme miteinander vernetzen und somit eine Vielzahl von proprietären Schnittstellen schaffen. Hierdurch entstehen den Unternehmen hohe Kosten, da mit Umstellungen an einem System gleichzeitig auch die Versorgung der



	Sachnr.	Benennung	TE Contr.	
	A 389 260 00 52	ZB SCHEIBE		
	A 389 262 49 35	GLEICHL.KOERPER		
	A 389 262 54 34	GLEICHLAUFRING		
	A 389 268 00 13	RASTHEBEL		
	A 389 268 00 51	ABSTANDRING		
	A 656 260 01 37	ZB RASTHEBEL		
	A 945 260 02 73	ZB SCHALTSEGMENT		
	A 945 260 03 57	ZB VENTIL GETRIEBESCHUTZ		
				1/17

Abb. 7.5: Datenkonsistenzprüfung durch Systemvergleiche

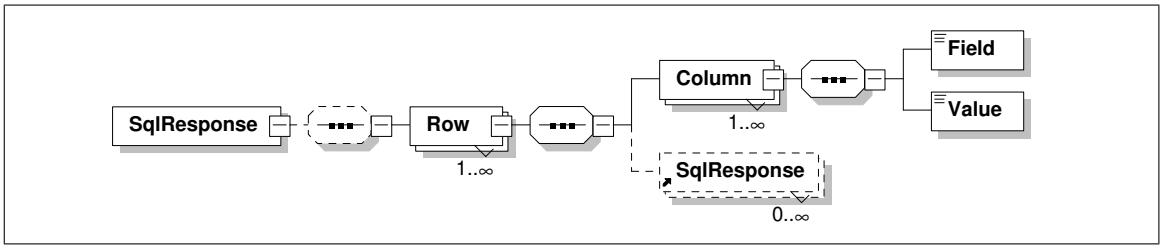


Abb. 7.6: Allgemeingültiges XML-Schema für SQL-Ergebnisse

abhängigen Systeme überprüft und teilweise angepasst werden. Der Trend geht bei dieser Thematik in Richtung von Universalschnittstellen, die mehrere Systeme versorgen können. XML ist dabei momentan Stand der Technik [CZR03].

Während der Realisierung des Simulationsdatenframeworks haben sich bereits Anwendungsfälle außerhalb des Simulationsumfelds ergeben. Es hat sich gezeigt, dass verschiedene Systeme auf die Echtzeit-Versorgung der simulationsrelevanten Daten zurückgreifen können. Durch die in Abschnitt 4.5.5 beschriebenen Konfigurationsmöglichkeiten müssen nicht zwangsläufig nur für die Simulation erforderliche Daten über das Framework bereitgestellt werden. Alle über Anbindungsmodule aus Abschnitt 4.5.3 verbundene Systeme können durch eine variabel gestaltbare Sicht mit den Notwendigen Daten versorgt werden. Hierfür muss jedoch das Darstellungsschema aus Abb. 4.10 abgeändert werden. Das Schema zur standardisierten Beschreibung von Simulationsmodellen wird durch ein flexibles, jedoch auch weniger aussagekräftigeres ersetzt (vgl. Abb. 7.6). Dabei werden nur noch Feldnamen und Feldinhalt Zeile für Zeile vom Framework ausgegeben. Mit diesem Schema kann jedes Ergebnis einer SQL-Abfrage in XML umgewandelt werden.

Die Simultaneous Engineering Werkzeuge zur integrierten Produktionsplanung, wie sie in Abschnitt 4.2 dargestellt sind, müssen auch mit Daten aus der realen Fabrik versorgt werden. Sinn und Zweck dieser Systeme ist die Verbesserung und Beschleunigung der Planungsaufgaben durch eine übergreifende Planungsdatenbank. Für die Versorgung dieser Datenquelle mit rea-

len Daten wurde in Zusammenarbeit mit der Firma Delmia eine XML-Schnittstelle auf das in dieser Arbeit entwickelte Datenframework entwickelt. Diese ermöglicht das Einlesen von planungsrelevanten Daten und soll künftig auch die in diesem System geplanten Zustände an die Produktivsysteme über das Framework zurückzuschreiben.

Ändert sich künftig ein über das Framework angebundenes System, so muss nur eine einzige Schnittstelle angepasst werden. Alle Systeme, die daraus mit Daten versorgt werden, werden die Daten in der genau gleichen Form erhalten, wie vor dem Systemwechsel.

7.4 Technische Grenzen

Im Praxiseinsatz hat sich gezeigt, dass sowohl die Datenzusammenführung als auch die automatisierte Modellgenerierung technisch möglich ist. Anhand der bisherigen Erfahrungen im Praxistest haben sich neue Anwendungsfelder und Möglichkeiten zur Weiterentwicklung ergeben. Es gibt jedoch auch Einschränkungen, die es in Zusammenhang mit den Praxiserfahrungen zu nennen gilt.

7.4.1 Einschränkungen bei der automatisierten Modellgenerierung

Eine der Zielsetzungen dieser Arbeit war das teilautomatisierte Generieren von Simulationsmodellen aus Produktionsdatensystemen. Das in Kapitel 4 entwickelte und in Kapitel 6 realisierte Konzept erfüllt diese Anforderung. Nach Fertigstellung des ersten Prototypen für das Datenframework und dem Parser zur Modellgenerierung in Quest wurden erste Praxistests durchgeführt. Hierfür wurde zuerst eine XML-Datei generiert, welche die gleiche Struktur wie das Ergebnis aus der Echtzeit-Anbindung aufweist, jedoch stark vereinfacht ist. In der Datei wurden 4 Maschinen, 3 Teile und 7 Prozesse mit allen zugehörigen Daten, wie Zeiten, Ausfälle, Stückzahlen, etc. hinterlegt. Hierdurch konnte der Parser und das von diesem generierte Quest-Modell verifiziert werden.

Mit der Umstellung auf das Framework hat sich gezeigt, dass auch diese Daten problemlos in ein Modell umgesetzt werden, nur der Ablauf in diesem war nicht valide. Da die Zusammenführung der Daten keine Fehler produziert hat, konnte die Datenqualität als Verursacher identifiziert werden. Selbst kleine Unstimmigkeiten in der Datenbasis führen oftmals zu einem nicht ausführbaren Modell. Die Menge an simulationsrelevanten Daten, die im Modell erzeugt wird, ist aber eine sehr gute Ausgangsbasis für die weitere Modellierung. Es muss nicht mehr jede Prozesszeit, Verbindung und Position von Hand modelliert werden. Allerdings kann die Suche nach Fehlern im Modell durch die übertragenen Daten keinem Laien bzgl. Simulation zugemutet werden. Somit kann auch kein Produktionsplaner quasi per Knopfdruck eine Simulationsstudie durchführen, ohne jegliche Kenntnisse über diese Methodik zu besitzen.

Ein passabler Zwischenschritt, der es letztendlich jedem Anwender erlaubt, Simulation ohne Vorkenntnisse zu bedienen, ist die Möglichkeit, die Daten aus den Produktivsystemen an die gewünschte und zu bewertende Alternative anzupassen. Hierzu speichert man ein Modell aus dem Datenframework in einer XML-Datei ab und überarbeitet es, bis es für die Simulation

valide ist. Diese Datei kann mittels der Oberfläche aus Abb. 6.16 in Abschnitt 6.3.2 bearbeitet werden. Darin können sämtliche Simulationsparameter verändert und auch als entsprechende Alternativzustände abgespeichert werden. Desweiteren kann daraus ein Modell in den genannten Simulatoren erzeugt werden.

7.4.2 Einschränkungen des Simulationsdatenframeworks

Die Systematik des Datenframeworks baut, abgesehen von den in Abschnitt 6.1.3.2 genannten Batch-Jobs, rein auf permanenten Netzwerkverbindungen zu Systemen auf. Dadurch ist es jedoch nur so leistungsfähig wie das langsamste System in der Kette der zusammenzuführenden Datenbanken. Es ist somit vergleichbar mit einer sequentiellen Fertigung. Fällt auch nur eine Ressource aus, so steht die gesamte Linie. In der Praxiserprobung trat dieses Phänomen zwar sehr selten auf, bei einem Auftreten kann jedoch die Anfrage nicht mehr vollständig beantwortet werden, obwohl vier von fünf Systemen noch Daten liefern. Ursache hierfür ist die Zusammenführung in eine XML-Struktur, bei der möglicherweise Schlüsselelemente in der Baumstruktur aufgrund des Ausfalls eines Systems fehlen.

Ein weiterer, schwer zu behandelnder Schwachpunkt ist die Rekursivität von Abfragen, welche auf der in Abschnitt 6.1.2.2 beschrieben Execution Engine auftreten können. Hierdurch wird eine Abschätzung der zu erwartenden Datenmenge annähernd unmöglich. Innerhalb des Datenframeworks kann durch eine Speicherüberwachung relativ einfach ein Absturz des Webservers verhindert werden. Auf den entfernten DBMSen lässt sich das jedoch nicht realisieren. Deshalb muss bei der Erstellung der Sichten sehr genau die zu erwartende Datenmenge abgeschätzt werden. Die in Abschnitt 3.4.2 beschriebene Parallelisierung der Abfrageausführungen mittels Thread-Pools kann bei unpassender Auslegung zu einem *Denial of Service* (Dienstverweigerung) beim Datenbanksystem führen, weil hunderte von Anfragen gleichzeitig gestellt werden. Allerdings hat sich im Praxistest gezeigt, dass eine Simulationsanfrage bis zu zwei Minuten benötigt, wenn die Abfragen rein sequentiell abgearbeitet werden. Im Vergleich hierzu wurden bei der Parallelisierung für die gleiche Abfrage nur ca. 10 Sekunden benötigt[1].

7.5 Schlussfolgerungen

Das Simulationssystem als Werkzeug zur Unterstützung der Produktionsplanung und kontinuierlichen Verbesserung der Fabrik ist eine Anwendung für Experten. Man benötigt viel Erfahrung, um leistungsfähige valide Modelle zu Erstellen und die Ergebnisse daraus richtig zu interpretieren. In Gesprächen mit Simulationsanwendern aller Bereiche zeigt sich immer wieder, dass kein Modell wie das andere ist. Sie sind einzigartige Konstrukte, die mit dem Hintergrund der Fragestellungen, die damit zu beantworten sind, erzeugt worden sind. Im Alltag gelingt es gelegentlich, ein standardisiertes Grundmodell für ähnliche Zielsetzungen und Abläufe zu schaffen.

[1] Testfall: Abfrage der Daten einer gesamten Kostenstelle, bestehend aus 58 Maschinen mit 3095 Fertigungsplänen und 1161 betroffenen Teilen. Ergebnisgröße: 1054 KiloByte

Auch hier liegen jedoch meistens die Schwierigkeiten im Detail, so dass bestimmte Anpassungen vorgenommen werden müssen. Der Modellierungsaufwand bei Nutzung des entwickelten Frameworks ist jedoch wesentlich geringer.

7.5.1 Nutzen in der praktischen Anwendung

Viele Anwendungsfälle der Materialflusssimulation in einer Teilefertigung basieren auf einem standardisierten Ablauf, wie er in dieser Arbeit geschildert wird. Das Zerlegen von Fertigungsbereichen in eine Matrixstruktur (Abschnitt 5.2.1), die Aggregierung zu einem Verbundnetzwerk (Abschnitt 5.2.2) und die standardisierte Steuerungslogik (Abschnitt 5.2.3) bilden Grundbausteine, die sich in fast jedem Modell wiederfinden.

Die automatische Versorgung dieses Modellgerüsts ist die logische Folgerung zur Erreichung eines Anwendungsstandards und zur festen Integration von Simulation in den Planungsprozess. Das modulare Datenframework erlaubt es, dass auch nur einzelne Bestandteile aus der Gesamtmenge an Daten, wie eine Maschine mit zugehörigen Teilen und Prozessen, geladen werden können. Diese können mit dem Parser für das entsprechende Simulationssystem direkt in ein vorhandenes oder manuell erstelltes Modell geladen werden. Diese Technik wird momentan am häufigsten genutzt. Einerseits wird so die Validierung der übertragenen Daten einfacher als bei einem komplett generierten Modell, andererseits kann dadurch quasi wie mit einem Baukastensystem modelliert werden. Der Simulationsexperte spart Zeit durch die automatische Generierung von Verteilungen für Ausfälle und Reparaturen. Es müssen folglich weniger Gespräche und Abstimmungen mit Instandhaltern und Meistern geführt werden.

Die Verbesserung der Ausgangsdaten aus den Systemen wurde mit den Anwendungen zur Datenkonsistenzprüfung aus Abschnitt 7.2 bereits in die Wege geleitet. Je besser die Qualität und Genauigkeit der Quelldaten, desto realitätstreuer werden auch die erzeugten Modelle. Außerdem verringert sich dann die Zeit für die Nachbearbeitung. Es muss sich jetzt ein Zustand einpendeln, der sowohl für die Simulation als auch für die Fabrik tragbar ist. Der Mensch kann durch Erfahrung und Kombinationsvermögen auch aus ungenaueren Daten die richtigen Schlüsse ziehen. Die Simulation kann dies nicht. Daher muss ein Mittelweg gefunden werden zwischen exakteren Daten (= höherer Zeitaufwand) und der Verbesserung bei der teilautomatisierten Modellgenerierung (= Einsparung durch Simulation). Die übergreifende Planungssoftware, wie von den Firmen Delmia und Tecnomatix, erzeugen exaktere Planungsdaten, als dies von Hand bei gleichem Zeitaufwand möglich wäre. Daher bleibt abzuwarten, wie sich diese Systeme über den Piloteinsatz hinaus etablieren und wie sie die Planung der Zukunft verändern werden. Die Technik zur Nutzung der in diesen Systemen generierten Daten ist mit dem Simulationsdatenframework geschaffen und kommt auch schon zum Einsatz.

7.5.2 Weiterentwicklung und zukünftige Einsatzfelder

Die Zusammenführungstechnik wird nun einerseits in Bezug auf die Toleranz bei unstimmigen Eingangsdaten und andererseits in Richtung der digitalen Planungsanwendungen weiterentwickelt. Im Rahmen der Methoden und Software der digitalen Fabrik steht ein viel breiteres Spek-

trum an Informationen zur Verfügung, als es während dieser Arbeit bei den konventionellen Planungsmethoden der Fall war. Hierzu zählen 3D-Geometrien für Teile, Produkte, Ressourcen und Layout, sowie Vorranggraphen, Logistikumfänge und MTM-Betrachtungen, um nur einige zu nennen. Diese gilt es jetzt für die Simulation zu nutzen und die Methode Simulation über den gesamten Planungsprozess hinweg fest zu integrieren.

Die im Rahmen dieser Arbeit entwickelte, standardisierte Modellbeschreibung in XML ist noch beliebig Erweiterbar. Die momentane Abdeckung einer beliebigen Teilefertigung ist dabei erst der Anfang. Ergänzungen für die Montage und den Logistikverkehr müssen folgen. Es wäre wünschenswert, dass aus diesem Ansatz künftig ein einheitlicher Standard wird, möglicherweise in Form einer VDI-Richtlinie. Im Bereich des CAD-Datenaustauschs sind bereits Vereinheitlichungen realisiert (z.B. DXF, STEP oder IGES) und wird von den gängigen CAD-Systemen unterstützt. Ebensolches ist für die Simulation erstrebenswert und würde die Arbeit für viele Simulationsexperten sowie die Integration von Planungssystemen sehr erleichtern. Eine Arbeitsgruppe von Simulationsanwendern aus unterschiedlichen Bereichen und Werken der Daimler-Chrysler AG hat bereits begonnen, formale Beschreibungen für Modelle auszuarbeiten. Dabei besteht ein Kernpunkt in der Definition von notwendigen und optionalen Parametern für verschiedene Entitäten eines Modells.

Durch den Einsatz der Online-Verbindungen zu diversen Produktivsystemen wird es möglich auf Leitstände zuzugreifen. Der aktuelle Status von Montagestationen, die Positionen bestimmter Getriebe im Montagesystem und die zu fertigenden Aufträge werden alle in diesem Erfasst. Daher laufen gerade die Vorbereitungen zur Anbindung des Montageleitstands an ein sehr präzises Simulationsmodell mit dem Ziel ein simulationsbasiertes Frühwarnsystem aufzubauen. Hierzu sollen in regelmäßigen Abständen Daten über das Simulationsdatenframework an das Modell übertragen werden, welches dadurch den aktuellen Zustand im Realsystem nachbilden kann. Anschließend sollen, abgekoppelt vom Realzustand, Prognosen für den weiteren Verlauf und Handlungsalternativen im Modell bewertet und ausgegeben werden.

Ein weiterer interessanter Aspekt der Simulation ist die ganzheitliche Bilanzierung . Während für gewöhnlich die Schwerpunkte bei den Materialflüssen, Ressourcenbelegungen, etc. liegen, so werden diese beim ganzheitlichen Ansatz noch um Medienströme erweitert [HJ05]. Hierzu zählen die Versorgung der Ressourcen mit Strom, Druckluft, Kühl- und Schmiermittel, etc.. Die hierfür benötigten Kenndaten, wie Anschlussleistung und Kühlmitteldurchfluss, sind ebenfalls in Planungs- und Dokumentationssystemen verfügbar und könnten künftig über das Datenframework herangezogen werden. Damit könnten während der Planungsphase schon Versorgungseinrichtungen betrachtet und neue Ressourcen dahingehend verbessert werden.

Kapitel 8

Zusammenfassung und Ausblick

Die Produktionsplanung in der Automobilindustrie erhebt heute den Anspruch, kein Werk und keine Anlage in Betrieb zu nehmen, ohne diese vorher mittels digitaler Planungsmethoden abgesichert zu haben [SS02]. Die Materialflusssimulation als eine Komponente der digitalen Planungsmethoden schafft dabei die Möglichkeit dynamische Abläufe zu analysieren und somit die beste Variante unter den Planungsalternativen zu selektieren.

Der Einsatz der Methode Materialflusssimulation ist zeitaufwändig und wird im klassischen Fall von einem Simulationsexperten als Dienstleistung für die Produktionsplanung erbracht. Bis die simulationsrelevanten Daten in den Planungssystemen und bei den Planern abgefragt und in ein Modell überführt sind, haben sich diese häufig schon wieder verändert. Es entsteht ein zeitlicher Verzug, welcher häufig durch wiederholte Anpassungszyklen angeglichen werden muss. Das hatte zur Folge, dass die Simulation nicht vollständig in den Planungsprozess integriert werden konnte. Hinzu kommt, dass die Methode nur einem kleinen Kreis von Simulationsexperten vorbehalten ist und somit nicht zu einem festen Bestandteil der Planung gemacht werden kann.

Im Rahmen dieser Arbeit wurde der Planungsprozess in Bezug auf die Anforderungen für die Integration von Simulation in diesen untersucht. Die Datenverarbeitungssysteme in den Industriebetrieben spielen dabei eine zentrale Rolle, da sehr viele der für die Simulation notwendigen Daten darin in elektronischer Form vorliegen. Bei konsequenter Umsetzung des Simultaneous Engineering arbeiten verschiedene Personen parallel an einem Planungsprojekt und befüllen gleichzeitig unterschiedliche Dokumentations- und Planungssysteme mit ihren Erkenntnissen. Daher stand zu Beginn dieser Arbeit die Idee, Daten aus den betrieblichen Datenverarbeitungssystemen online zur teilautomatisierten Generierung von standardisierten Simulationsmodellen zu nutzen. Diese können von Simulationsexperten weiter ausgebaut und den Anforderungen entsprechend angepasst werden.

Simulationsrelevante Daten sind über verschiedene Systeme verteilt. Hierzu zählen Datenbanken, Hostsysteme und Netzlaufwerke verschiedener Softwarehersteller und Architekturen. Zur Zusammenführung von Daten aus diesen unterschiedlichen Systemen wurde ein Datenframework auf der Basis von Web-Diensten realisiert, welches über Anbindungsmodule mit den einzelnen Datenquellen verbunden ist. Dieses beinhaltet eine Logik, welche es ermöglicht, Daten flexibel zusammenzustellen und in einer standardisierten Form auf Basis von XML darzustellen.

Die XML-Daten, welche während der Anfrage mit den aktuellen Planungsdaten generiert werden, können auf einem Clientrechner grafisch dargestellt und nachbearbeitet werden. Der Anwender hat die Möglichkeit die Daten auf Korrektheit und Übertragbarkeit auf ein Simulati-

onsmodell zu überprüfen. Es wird weiterhin die Möglichkeit geschaffen, auf Basis des aktuellen Planzustands verschiedene Alternativen abzuleiten. Diese überarbeiteten Alternativen können anschließend mittels XML-Parsern in den Simulationssystemen Quest, Automod und Simflex/3D erzeugt, ausgeführt und analysiert werden. Die daraus gewonnenen Erkenntnisse können schließlich wieder in den Planungsprozess miteinbezogen werden.

Die in dieser Arbeit beschriebene Methodik ermöglicht es Planungsmitgliedern mit Grundkenntnissen in der Simulation diese für einfache Fragestellungen, wie Kapazitätsanalysen, Losgrößenbestimmung und Pufferdimensionierung, zu nutzen. Außerdem wird auch für den Simulationsexperten der Zeitaufwand für die Datensammlung, -aufbereitung und die Modellierung reduziert.

Im Rahmen dieser Arbeit wurden vorerst nur die notwendigen Grunddaten für den Aufbau eines Modells betrachtet. Die Web-Dienste können modular erweitert werden und die Beschreibung von Modellen in XML ist beliebig erweiterbar. Durch die konsequente Weiterentwicklung dieser Methode kann zukünftig eine standardisierte Modellbeschreibung entwickelt werden, welche von unterschiedlichen Simulationssystemen lesbar ist. Die Tatsache, dass die Daten von dem entwickelten Framework online bereitgestellt werden, ermöglicht künftig die Kopplung von Leitstandsystemen an vorhandene Simulationsmodelle mit dem Ziel, die Modelle auch nach Serienstart weiterzuverwenden und als Frühwarnsysteme einzusetzen.

Literaturverzeichnis

[Abe03] ABECK, Sebastian: *Verteilte Informationssysteme*. dpunkt-Verl., 2003. – ISBN 3–89864–188–0 26

[ABS00] ABITEBOUL, S. ; BUNEMAN, P. ; SUCIU, D.: *Data on the Web - From Relations to Semistructured Data and XML*. Morgan Kaufmann Publishers, 2000 51, 53

[Acè92] ACÈL, P. P.: Systems-Engineering und Simulation. In: *Tagungsband „Simulation – Ein Blick in die Zukunft“* BWI-IFOR der ETH Zürich, 1992 xi, 2

[Aßm02] ASSMANN, J.: Digitale Fabrik – Vision oder Realität. In: NOCHE, B. (Hrsg.) ; WITT, G. (Hrsg.): *Anwendungen der Simulationstechnik in Produktion und Logistik. 10. ASIM-Fachtagung*. Erlangen, 2002 12

[Ama94] AMANN, W.: *Eine Simulationsumgebung für Planung und Betrieb von Produktionssystemen*. Berlin : Springer, 1994 xi, 16

[AQM+97] ABITEBOUL, S. ; QUASS, D. ; MCHUGH, J. ; WIDOM, J. ; WIENER, J. L.: The Lorel Query Language for Semistructured Data. In: *International Journal on Digital Libraries* 1 (1997), Nr. 1, S. 68–88 53

[Arn04] ARNOLD, D.: *Handbuch Logistik*. 2. Berlin, Heidelberg : Springer Verlag, 2004 xi, 47

[Bac96] BACKES, M.: *Simulationsunterstützung zur zielorientierten Produktionsprozessplanung und -regelung*, Universität Mannheim, Diss., 1996 xi, 7, 12, 13, 15

[Ban00] BANKS, J.: Simulation in the Future. In: JOINES, J. A. (Hrsg.) ; BARTON, R. R. (Hrsg.) ; KANG, K. (Hrsg.) ; FISHWICK, P. A. (Hrsg.) ; IEEE (Veranst.): *Proceedings of the 2000 Winter Simulation Conference*. Piscataway, New Jersey, 2000, S. 1568–1576 54

[Bar03] BARNEKOW, T.: *Ein regelbasierter Beobachter zur prozessorientierten Integration betrieblicher Informationssysteme*, Universität Stuttgart, Fak. Maschinenbau, Institut für Arbeitswissenschaft und Technologiemanagement, Dissertation, 2003 22

[BB03] BRACHT, U. ; BERGBAUER, J.: Digitale Fabrikplanung in einer virtuellen Umgebung. In: SCHULZE, T. (Hrsg.) ; SCHLECHTWEG, S. (Hrsg.) ; HINZ, V. (Hrsg.): *Simulation und Visualisierung 2003*. Erlangen : SCS, 2003, S. 3–8 19

[BCNN00] BANKS, J. ; CARSON, J. S. ; NELSON, B. L. ; NICOL, D.: *Discrete-Event System Simulation*. 3. New Jersey : Prentice-Hall, 2000 39

[Ber00] BERNARD, B.: *Integration of manufacturing simulation tools with information sources*. Schweden, Royal Institute of Technology, Stockholm, Licentiate Thesis, 2000 39, 40

[Bla00] BLAZEJEWSKI, G.: *Produktionssteuerung mittels modularer Simulation*. Chemnitz : Gesellschaft für Unternehmensrechnung und Controlling, 2000 10

[Bol04] BOL, Georg: *Deskriptive Statistik*. 6. Oldenbourg, 2004. – ISBN 3–486–57612–7 xii, 101, 106

[Bro02] BROCKHAUS: *Der Brockhaus in fünfzehn Bänden*. 2. Mannheim : F.A. Brockhaus, 2002 8

[Bül95] BÜLTZINGSLOEWEN, G. von: CORBA Based Interoperability of Tools and Services in Engineering Environments. In: FARIA, L. (Hrsg.): *Proc. of the Int. Workshop on Concurrent/Simultaneous Engineering Frameworks and Applications*. Lisboa, Protugal, February 1995, S. 361–373 32

[BWS02] BLATECKY, A. ; WEST, A. ; SPADA, M.: Middleware the new frontier / EDUCAUSE review July/August. 2002. – Forschungsbericht 57, 58

[C+03] CASTANO, S. u. a.: Data Schema Integration in Web-Enabled Systems. In: DAHANAYAKE, A. (Hrsg.) ; GERHARDT, W. (Hrsg.): *Web-Enabled Systems Integration*. London : Idea Group Publishing, 2003, Kapitel 3., S. 41–65 33

[CL54] CHERNOFF, H. ; LEHMANN, E. L.: The use of maximum likelihood estimation in χ^2 tests for goodness of fit. In: *Annals of Mathematical Statistics* 25 (1954), S. 579–586 102

[Con97] CONRAD, S.: *Föderierte Datenbanksysteme – Konzepte der Datenintegration*. Berlin : Springer, 1997 xi, 28

[CT96] CHEN, M.-C. ; TSAI, D.-M.: Simulation optimization through direct search for multi-objective manufacturing systems. In: *Production Planning & Control* 7 (1996), Nr. 6, S. 554–565 13

[CZR03] CHAUDHRI, A. ; ZICARI, Roberto ; RASHID, Awais: *XML Data Management: Native XML and XML Enabled DataBase Systems*. Boston, MA, USA : Addison-Wesley, 2003. – ISBN 0201844524 125

[D+98] DEUTSCH, A. u. a.: XML-QL: A Query Language for XML / W3C. Version: 1998. `http://www.w3.org/TR/NOTE-xml-ql`. 1998. – Forschungsbericht 33

[Dad96] DADAM, P.: *Verteilte Datenbanken und Client/Server-Systeme*. Berlin : Springer, 1996 21

[DBE98] DELEN, D. ; BENJAMIN, P. C. ; ERRAGUNTLA, M.: Integrating modeling and analysis generator environment (IMAGE): a decision support tool. In: MEDEIROS, D. J. (Hrsg.) ; WATSON, E. F. (Hrsg.) ; CARSON, J. S. (Hrsg.) ; MANIVANNAN, M. S. (Hrsg.): *Proceedings of the 1998 Winter Simulation Conference*. Washington, D. C. : IEEE Computer Society Press, 1998, S. 1401–1408 40

[DBE99] DELEN, D. ; BENJAMIN, P. C. ; ERRAGUNTLA, M.: An integral toolkit for enterprise modeling and analysis. In: FARRINGTON, P. A. (Hrsg.) ; NEMBHARD, H. B. (Hrsg.) ; STURROCK, D. T. (Hrsg.) ; EVANS, G. W. (Hrsg.): *Proceedings af the 1999 Winter Simulation Conference* Bd. 1. Phoenix, Arizona : ACM Press, 1999, S. 289–297 40

[DD02] DOMSCHKE, W. ; DREXL, A.: *Einführung in Operations Research*. 5. Berlin : Springer, 2002 7, 10, 11, 71, 72

[Deh03] DEHNHARDT, W.: *Java und Datenbanken: Anwendungsprogrammierung mit JDBC, Servlets und JSP*. München : Hanser, 2003 xi, 35

[Deu94] DEUTSCH, J. M.: The evolution of customer middleware requirements. In: *Proceedings of the Third International Conference on Parallel and Distributed Information Systems*, 1994 57

[DG92] DALLERY, Y. ; GERSHWIN, S. B.: Manufacturing flow line systems: a review of models and analytical results. In: *Queueing Systems* 12 (1992), S. 3–94 100

[DIN87] DIN: *69900 Netzplantechnik*. Deutsche Industrienorm 69900. Berlin: Beuth Verlag, 08 1987 81

[DP88] DÖRFLER, W. ; PESCHEK, W.: *Einführung in die Mathematik für Mathematiker*. München : Hanser, 1988 11

[Edw99] EDWARDS, Jeri: *3-tier client-server at work*. New York, Weinheim : Wiley, 1999 54

[EGP99] ECKARDT, F. ; GMILKOWSKY, P. ; PALLEDUHN, D.: Konzeption und Entwicklung eines objektorientierten und wissensbasierten Simulationssystems zur automatischen Generierung von Simulationsmodellen diskreter Produktionsprozesse. In: DESSEN, O. (Hrsg.) ; LORENZ, P. (Hrsg.): *Simulation und Visualisierung '99, Proceedings der Tagung „Simulation und Visualisierung '99" der Otto-von-Guericke-Universität Magdeburg*. Magdeburg, 1999, S. 225ff 40

[Erl04] ERL, T.: *Service-oriented architecture: a field guide to integrating XML and Web services*. Upper Saddle River, NJ, USA : Prentice Hall, 2004 86

[FH99] FONG, J. ; HUI, R.: Application of middleware in the three tier client/server database design methodology. In: *Journal of the Brazilian Computer Society* 6 (1999), Nr. 1, S. 50–64 55

[Fis97] FISHWICK, P. A.: Web-based simulation. In: *WSC '97: Proceedings of the 29th conference on Winter simulation*. New York, NY, USA : ACM Press, 1997. – ISBN 0–7803–4278–X, S. 100–102 40

[Fis01] FISHMAN, G. S.: *Discrete event simulation*. New York : Springer, 2001 11

[FPW90] FRAUENSTEIN, T. ; PAPE, U. ; WAGNER, O.: Objektorientierte Sprachkonzepte und diskrete Simulation. In: MÖLLER, D. (Hrsg.) ; SCHMIDT, B. (Hrsg.): *Fachberichte Simulation* Bd. 13. Berlin : Springer, 1990 11

[FR00] FELDMANN, K. ; REINHART, G.: *Simulationsbasierte Planungssysteme für Organisation und Produktion*. Berlin : Springer, 2000 16

[Fra99] FRATERNALI, P.: Tools and approaches for developing data-intensive Web applications: a survey. In: *ACM Computing Surveys* 31 (1999), Nr. 3, S. 227–263 xi, 55

[Fre04] FRETER, Hermann: *Marktsegmentierung*. 2. Kohlhammer, 2004. – ISBN 3–17–018319–2 16

[GB05] GRAUPNER, T.-D. ; BIERSCHENK, S.: Erfolgsfaktoren bei der Einführung der Digitalen Fabrik. In: *Industrie Management* 21 (2005), Nr. 2, S. 59–62 xi, 17, 44

[GEP97] GMILKOWSKY, P. ; ECKARDT, F. ; PALLEDUHN, D.: Concept of an integrated system for modeling, simulation and analysis of discrete production processes. In: OBAIDAT, M. S. (Hrsg.): *Porceedings of the 1997 summer Computer Simulation Conference*, SCSI, 1997, S. 329–334 40

[GEP98] GMILKOWSKY, P. ; ECKARDT, F. ; PALLEDUHN, D.: Automated simulation model generation: a knowledge-based approach within an integrated system for modeling, simulation and analysis of discrete production processes. In: ADIS, M. (Hrsg.) ; GRIEBENOW, R. (Hrsg.) ; Society for Computer Simulation (Veranst.): *Proceedings of the Simulators International XV Advanced Simulation Technologies Conference AST* Society for Computer Simulation, 1998, S. 157–162 40

[Gor61] GORDON, G.: A general purpose systems simulation program. In: *Proceedings of the Eastern Joint Computer Conference*. Washington, D.C., 1961 7

[Gor03] GORA, H. J.: Einsatzfelder der Simulation in der Automobilindustrie. In: BAYER, J. (Hrsg.) ; COLLISI, T. (Hrsg.) ; WENZEL, S. (Hrsg.): *Simulation in der Automobilproduktion*. Berlin : Springer, 2003, S. 17–27 14

[Gra03] GRABS, T.: *Storage and Retrieval of XML Documents with a Cluster of Database Systems*. Zürich, ETH Zürich, Diss., 2003 51

[GRS02] GRAUPNER, T.-D. ; RICHTER, H. ; SIHN, W.: Web-based simulation 2: configuration, simulation and animation of manufacturing systems via the internet. In: *WSC '02: Proceedings of the 34th conference on Winter simulation*, Winter Simulation Conference, 2002. – ISBN 0–7803–7615–3, S. 825–831 xi, 40, 41

[Gru99] GRUNDIG, C.-G.: Simulationstechnik sichert Transparenz des Investitionsobjektes. In: *ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb* 94 (1999), Nr. 5, S. 273–277 13

[GT00] GÜNTHER, H.-O. ; TEMPELMEIER, H.: *Produktion und Logistik*. 4. Berlin : Springer, 2000 xii, 73

[H+99] HAAS, L. u. a.: Transforming heterogeneous data with database middleware: Beyoundintegration. In: *IEEE Data Engineering Bulletin* 22 (1999), Nr. 1, S. 31–36 33

[Hac93] HACKATHORN, R. D.: *Enterprise database connectivity*. New York : Wiley, 1993 23

[Har98] HARTUNG, J.: *Statistik: Lehr- und Handbuch der angewandten Statistik*. 11. München : Oldenbourg, 1998 101, 104

[Hat01] HATEM, V.: Integrating Capacity Simulation into Process Planning. In: PETERS, B. A. (Hrsg.) ; SMITH, J. S. (Hrsg.) ; MEDEIROS, D. J. (Hrsg.) ; ROHRER, M. W. (Hrsg.): *Proceedings of the 2001 Winter Simulation Conference*, ACM Press, 2001, S. 1470ff 53

[Hei97] HEIM, J. A.: Integrating distributed simulation objects. In: *WSC '97: Proceedings of the 29th conference on Winter simulation*. New York, NY, USA : ACM Press, 1997. – ISBN 0–7803–4278–X, S. 532–538 40

[HJ05] HESSELBACH, H. ; JUNGE, M.: Redurzierung von Energiespitzen durch Fabriksimulation. In: *Industrie Management* 21 (2005), Nr. 2, S. 35ff 129

[HJW03] HELLMANN, A. ; JESSEN, U. ; WENZEL, S.: e-Services - a part of the „Digital Factory“. In: *Proceedings of 36th CIRP-International Seminar on Manufacturing Systems, Universität des Saarlandes, 03.*- Bd. 29. Saarbrücken, 2003 (Produktionstechnik) 15

[Hoo86] HOOPER, J. W.: Strategy-related characteristics of discrete-event languages and models. In: *Simulation* 46 (1986), Nr. 4, S. 153–159 11

[Hot05] HOTZ, I.: Optimierung der Prüfstandsbelegung in der Getriebeproduktion durch hierarchische Verknüpfung von Simulation und Optimierung. In: SCHULZE, T. (Hrsg.) ; HORTON, G. (Hrsg.) ; PREIM, B. (Hrsg.) ; SCHLECHTWEG, S. (Hrsg.): *Proceedings der Tagung „Simulation und Visualisierung 2005“ am Institut für Simulation und Graphik der Otto-von-Guericke-Universität Magdeburg*. Erlangen : SCS Publishing House, 2005, S. 329–337 40

[HTMS05] HANISCH, A. ; TOLUJEW, J. ; MEUSCHKE, T. ; SCHULZE, T.: Methoden zur Initialisierung von Online-Simulationsmodellen. In: HÜLSMANN, F. (Hrsg.) ; KOWARSCHICK, M. (Hrsg.) ; RÜDE, U. (Hrsg.): *Proceedings 189th Symposium Simulationstechnik ASIM 2005*. Erlangen : SCS Publishing House e.V., 2005, S. 388–393 41

[ISO86] ISO: *ISO 8879. Information Processing - Text and Office Systems - Standard Generalized Markup Language (SGML)*. 1986 51

[Jak91] JAKOBI, H. A.: Akzeptanzförderung „Simulationsanwendung“. In: *Simulation und Verstehen*, 1991 (ASIM-Fachtagung) 14

[JCKS04] JORDAN, M. ; CZAJKOWSKI, G. ; KOUKLINSKI, K. ; SKINNER, G.: Extending a J2EE server with dynamic and flexible resource management. In: *Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*. New York, NY, USA : Springer-Verlag NEW York, Inc., 2004, S. 439–458 57

[Joh01] JOHANSSON, M.: *Information management for manufacturing system development*. Sweden, Division of Computer Systems for Design and Manufacturing, Department of Production Engineering, Royal Institute of Technology, Stockholm, Ph. D. Thesis, 2001 40

[KG95] KOŠTURIAK, J. ; GREGOR, M.: *Simulation von Produktionssystemen*. Wien : Springer, 1995 8, 12

[Kil03] KILGORE, R. A.: Object-oriented simulation with SML and SILK in .NET and Java. In: CHICK, S. (Hrsg.) ; SANCHEZ, P. J. (Hrsg.) ; FERRIN, D. (Hrsg.) ; MORRICE, D. J. (Hrsg.): *Proceedings of the 35th conference on Winter simulation*, Winter Simulation Conference, 2003, S. 218–224 67

[KK00] KLAUS, Peter ; KRIEGER, Winfried: *Logistik: Grundlagen, Strategien, Anwendungen*. Berlin, Heidelberg, New York : Springer Verlag, 2000 46

[Kmu00] KMUCHE, W.: *Strategischer Erfolgsfaktor Wissen*. Köln : Dt. Wirtschaftsdienst, 2000 v

[Knu73] KNUTH, D.: *The Art of Computer Programming*. Bd. 3. Addison Wesley, 1973 59

[Kos99] KOSCHEL, A.: *Ereignisgetriebene CORBA-Dienste für heterogene, verteilte Informationssysteme*. Karlsruhe, Universität Karlsruhe, Diss., 1999 32

[KP00] KULJIS, J. ; PAUL, R. J.: A review of web based simulation: whither we wander? In: *WSC '00: Proceedings of the 32nd conference on Winter simulation*. San Diego, CA, USA : Society for Computer Simulation International, 2000. – ISBN 1–23456–789–0, S. 1872–1881 40

[Kra01] KRAUSE, F.-L.: Digitale Fabrik. In: *ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb* 96 (2001), Nr. 3, S. 84ff 12

[Kud92] KUDLICH, H.: *Verteilte Datenbanken*. Berlin : Siemens-Aktiengesellschaft, 1992 21

[KW97] KARREIS, A. ; WELLER, A.: Analyse und Optimierung komplexer Produktionswerkzeuge mit Simulationswerkzeugen. In: *ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb* 92 (1997), Nr. 10, S. 530–533 13

[LCG87] LOVE, D. M. ; CLARKE, S. R. ; GOODEN, D. I.: The use of simulation to determine operational policies in an MRP/FMS environment. In: MCGEOUGH, J. A. (Hrsg.): *Second International Conference on Computer-Aided Production Engineering*. Bury St. Edmunds, UK : Mechanical Engineering Purclications, 1987, S. 261–266 40

[LGW99] LINNER, S. ; GEYER, M. ; WUNSCH, A.: Optimierte Prozesse durch Digital Factory Tools / Optimizing Manufacturing Processes with Digital Factory Tools. In: *VDI Bericht 1489* 1 (1999), S. 187–198 xi, 42

[Lie95] LIEBL, F.: *Simulation: problemorientierte Einführung*. 2. München : Oldenbourg, 1995 11

[Lil67] LILLIEFORS, H. W.: On the Kolmogorov-Smirnov test for normality with mean and variance unknown. In: *Journal of the American Statistical Association* 62 (1967), S. 387–402 105

[LL95] LANG, S. M. ; LOCKEMANN, P. C.: *Datenbankeinsatz*. Berlin : Springer, 1995 xi, 21, 22, 24, 27, 29, 30

[LP98] LIYANAGE, K. ; PERERA, T.: Design and development of a rapid data collection methodology. In: *International Conference on Simulation '98* IEEE, 1998, S. 297–304 39

[LRO96] LEVY, A. ; RAJARAMAN, A. ; ORDILLE, J. J.: Querying heterogeneous information sources using source descriptions. In: *Proceedings of the 22nd International Conference on Very Large Data Bases (VLDB'96)*. Bombay, India, September 3-6 1996, S. 251–262 33

[Meh94] MEHL, H.: *Methoden verteilter Simulation*. Braunschweig : Vieweg, 1994 11

[MHDE05] MASIN, M. ; HERER, Y. T. ; DAR-EL, E. M.: Design of self-regulating production control systems by Tradeoffs Programming. In: *IIE Transactions* 37 (2005), March, Nr. 3, S. 217–232. http://dx.doi.org/10.1080/07408170590899616. – DOI 10.1080/07408170590899616 xii, 72, 74

[MHZL03] MCLEAN, C. ; HARRELL, C. ; ZIMMERMANN, P. M. ; LU, R. F.: Simulation Standards: Current status, needs, and future directions. In: CHICK, S. (Hrsg.) ; SÁNCHEZ, P. J. (Hrsg.) ; FERRIN, D. (Hrsg.) ; MORRICE, D. J. (Hrsg.): *Proceedings of the 2003 Winter Simulation Conference*, 2003, S. 2019–2026 45, 48, 53, 54

[Net05] NETCRAFT: Zitat Internet-Statistik-Service (http://www.netcraft.com). (2005). http://www.netcraft.com 87

[NM02] NEUMANN, K. ; MORLOCK, M.: *Operations Research*. München : Hanser, 2002 43, 80

[Noc90] NOCHE, B.: *Simulation in Produktion und Materialfluss*. Köln : TÜV Rheinland, 1990 11

[Orr98] ORR, K.: Data Quality and Systems Theory. In: *CACM* 41 (1998), Nr. 2, S. 66–71 124

[OS99] OBERWEIS, A. ; SNEED, H.M.: *Software Management 99*. Stuttgart : Teubner, 1999 xi, 9

[ÖV99] ÖZSU, T. M. ; VALDURIEZ, P.: *Principles of Distributed Database Systems*. 2. Englewood Cliffs, New Jersey : Prentice Hall, 1999 26, 27

[Pag91] PAGE, B.: *Diskrete Simulation: eine Einführung mit Modula-II*. Berlin : Springer, 1991 10, 11

[Pag98] PAGE, E. H.: The rise of Web-based simulation: implications for the high level architecture. In: *WSC '98: Proceedings of the 30th conference on Winter simulation*. Los Alamitos, CA, USA : IEEE Computer Society Press, 1998. – ISBN 0–7803–5134–7, S. 1663–1668 40

[PH72] PEARSON, E. S. ; HARTLEY, H. O.: *Biometrika tables for statisticians II*. London : Cambridge University Press, 1972 105

[PS98] PARENT, C. ; SPACCAPIETRA, S.: Database integration: An overview of issues and approaches. In: *Communications of the ACM* 41 (1998), Nr. 5, S. 166–178 33

[Rah94] RAHM, E.: *Mehrrechner-Datenbanksysteme*. 1. Bonn : Addison-Wesley, 1994 xi, 25, 27, 29

[RB01] RANDELL, L. G. ; BOLMSJÖ, G. S.: Database driven factory simulation: a proof-of-concept demonstrator. In: *WSC '01: Proceedings of the 33nd conference on Winter simulation*. Washington, DC, USA : IEEE Computer Society, 2001. – ISBN 0–7803–7309–X, S. 977–983 xi, 39, 40

[Red96] REDMAN, T.: *Data Quality for the Information Age*. Boston : Artech House, 1996 124

[Red98] REDMAN, T.: The Impact of Poor Data Quality on the Typical Enterprise. In: *CACM* 41 (1998), Nr. 2, S. 79–82 124

[REF91] REFA (Hrsg.): *Arbeitsgestaltung in der Produktion*. München : Carl Hanser, 1991. – (Methodenlehre der Betriebsorganisation) 18, 68

[Rei77] REINHARDT, A.: Entwurf von Materialflusssystemen und Experimentsteuerung mittels grafisch-interaktiver Simulation. In: *Informatik-Fachberichte* Bd. 11. Berlin : Springer, 1977 7

[Rei85] REINHARDT, A.: Realzeitsteuerung mit dem graphisch-interaktiven Simulator SIMFLEX/2. In: *Informatik-Fachberichte, ASIM 85* Bd. 109. Berlin, 1985, S. 512–516 40, 50

[Rei88] REINHARDT, A.: Designing the Layout and the Control System of FMS. In: *Proceedings of the NATO-ASIM-CIM Istanbul*. Berlin, 1988 xii, 107

[Rei89] REINHARDT, A.: Integration von Modellen. In: *Simulation und Integration – Tagungsbericht ASIM Arbeitskreis für Simulation in der Fertigungstechnik*. München, 1989 xii, 109

[Rei03] REINHARDT, A.: Geschichten zur Simulation mit der Automobilindustrie. In: BAYER, J. (Hrsg.) ; COLLISI, T. (Hrsg.) ; WENZEL, S. (Hrsg.): *Simulation in der Automobilproduktion*. Berlin : Springer, 2003, S. 7–16 8

[RG00] RAMAKRISHNAN, R. ; GEHRKE, J.: *Database management systems*. 2. Boston : McGraw Hill, 2000 23

[Rib00] RIBBENS, J. A.: *Simultaneous Engineering for New Product Development*. New York : John Wiley & Sons, 2000 14

[Rit98] RITTER, D.: The middleware muddle. In: *SIGMOD Rec.* 27 (1998), Nr. 4, S. 86–93 57

[RJ03] REINHARDT, A. ; JENSEN, S.: Integration industrieller DV-Systeme zur automatischen Modellgenerierung in der Getriebeproduktion. In: SCHULZE, T. (Hrsg.) ; SCHLECHTWEG, S. (Hrsg.) ; HINZ, V. (Hrsg.): *Simulation und Visualisierung 2003*. Erlangen : SCS, 2003, S. 337–348 xii, 20, 67, 69

[Rol03] ROLLAND, F. D.: *Datenbanksysteme im Klartext*. München : Pearson Education, 2003 24

[RVJ03] REINHARDT, A. ; VERZANO, N. ; JENSEN, S.: Formale Beschreibung von Simulationsmodellen in XML. In: HOHMANN, R. (Hrsg.): *Simulationstechnik - 17. Symposium in Magdeburg*. Erlangen : SCS-Europe, 2003, S. 69–74 xi, 48, 81

[Sac97] SACHS, L.: *Angewandte Statistik*. 8. Berlin : Springer, 1997 100, 106

[SB03] SCHMIDT, D. C. ; BUSCHMANN, F.: Patterns, frameworks and middleware: thier synergistic relationships. In: *Porceedings of the 25th international conference on Software engineering*, 2003 57

[SBH93] SCHEFSTRÖM, D. ; BROEK (HRSG.), G. van d.: *Tool Integration: Environments and Frameworks*. Chichester : John Wiley & Sons, 1993 (Wiley Series in Software Based Systems) 32

[Sch00a] SCHRAGE, M.: *Serious Play: How The Worlds's Best Companies Simulate to Innovate*. Boston : Harvard Business School Press, 2000 12

[Sch00b] SCHÖTTNER, Josef: Wieviel ERP braucht die Konstruktion? In: *CAD-World* 6 (2000), Nov./Dez., S. 84f 42

[Sch01] SCHNEIDER, S.: *Rechnergestützte, kooperativ arbeitende Optimierungsverfahren am Beispiel der Fabriksimulation*, Universität Kassel, Diss., 2001 13

[Sch03] SCHUMANN, M.: Implementierung von verteiltem Training unter Nutzung der High Level Architecture. In: HOHMANN, R. (Hrsg.): *Simulationstechnik - 17. Symposium in Magdeburg*. Erlangen : SCS-Europe, 2003, S. 59–64 13

[SJ97] SRINIVASAN, K. ; JAYARAMAN, S.: Integration of simulation with enterprise models. In: *WSC '97: Proceedings of the 29th conference on Winter simulation*. New York, NY, USA : ACM Press, 1997. – ISBN 0–7803–4278–X, S. 1352–1356 40

[SKS97] SILBERSCHATZ, A. ; KORTH, H. ; SUDARSHAN, S.: *Database System Concepts*. McGraw Hill, 1997 57

[SL90] SHETH, A. P. ; LARSON, J. A.: Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases. In: *ACM Computing Surveys* 22 (1990), Nr. 3, S. 183–236 29, 31

[SM01] SLY, D. ; MOORTHY, S.: Simulation Data Exchange (SDX) Implementation and Use. In: PETERS, B. A. (Hrsg.) ; SMITH, J. S. (Hrsg.) ; MEDEIROS, D. J. (Hrsg.) ; ROHRER, M. W. (Hrsg.): *Proceedings of the 2001 Winter Simulation Conference*, 2001 53

[SS83] SCHLAGETER, G. ; STUCKY, W.: *Datenbanksysteme: Konzepte und Modelle*. 2. Stuttgart : Teubner, 1983 22, 23

[SS02] SCHILLER, E. ; SEUFFERT, W.-P.: Digitale Fabrik / Strategie – Bis 2005 realisiert. In: *Automobil Produktion (Sonderdruck)* 2 (2002), S. 4–10 xi, 12, 15, 20, 131

[SZ92] SPEARMAN, L. M. ; ZAZANIS, M. A.: Push and Pull Production Systems: Issues and Comparisons. In: *Operations Research* 40 (1992), Nr. 3, S. 521–532 71

[Try94] TRYBULA, W.: Building simulation models without data. In: *1994 IEEE International Conference on Systems, Man and Cybernetics. Humans, Information and Technology* Bd. 1 IEEE, 1994, S. 209–214 39

[UJ97] UMEDA, S. ; JONES, A.: Simulation in Japan: State-of-the-art update / NISTIR 6040, National Institute of Standards and Technology, U.S. Department of Commerce, Technology Administration. 1997. – Forschungsbericht 39

[Ull97] ULLMAN, J. D.: Information Integration Using Logical Views. In: *Proceedings of the 6th International Conference on Database Theory*, 1997 xi, 36, 37

[ÜSC+99] ÜLGEN, O. M. ; SHORE, J. ; COFFMAN, G. ; SLY, D. ; ROHRER, M. ; WOOD, D.: Increasing the Power and Value of Manufacturing Simulation via Collaboration with other analytical Tools. In: FARRINGTON, P. A. (Hrsg.) ; NEMBHARD, H. B. (Hrsg.) ; STURROCK, D. T. (Hrsg.) ; EVANS, G. W. (Hrsg.): *Proceedings of the 1999 Winter Simulation Conference* Bd. 1. Phoenix, Arizona : ACM Press, 1999, S. 749–753 54

[VB05] VAHS, D. ; BURMESTER, R.: *Innovationsmanagement: von der Produktidee zur erfolgreichen Vermarktung*. 3. Stuttgart : Schäffer-Poeschel, 2005 xi, 14, 43

[VDI02] VDI: Die digitale Fabrik wird Chefsache im Automobilbau. In: *VDI Nachrichten* 28 (2002), S. 9 16

[VDI05] VDI: *VDI-Handbuch Materialfluss und Fördertechnik*. Bd. 8: Materialfluss II (Organisation/Steuerung). Düsseldorf : VDIVerl., 2005 8, 15

[VW02] VOSSEN, G. ; WITT, K.-U.: *Grundlagen der theoretischen Informatik mit Anwendungen*. 2. Braunschweig : Vieweg, 2002 63

[W3C98] W3C: Extensible Markup Language (XML) 1.0 / World Wide Web Consortium (T. Bray and J. Paoli and C. M. Sperberg-McQueen, Hrsg.). Version: February 1998. `http://w3.org/TR/1998/REC-xml-19980210`. 1998. – Forschungsbericht 51, 52

[W3C99] W3C: XML Path Language (XPath) Version 1.0 / World Wide Web Consortium (J. Clark, S. DeRose, Eds.). Version: November 1999. `http://www.w3.org/TR/1999/REC-xpath-19991116`. 1999. – Forschungsbericht 52

[W3C04] W3C: XML Schema / World Wide Web Consortium (D. C. Fallside, P. Walmsley, Eds.). Version: October 2004. `http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/`. 2004. – Forschungsbericht 52

[WBN96] WILLIAMS, T.J. ; BERNUS, P. ; NEMES, L.: The concept of enterprise integration. In: BERNUS, P. (Hrsg.) ; NEMES, L. (Hrsg.) ; WILIAMS, T. J. (Hrsg.): *Architectures for the Enterprise Integration*. London : Chapman and Hall, 1996 15

[WHP97] WHITMAN, L. ; HUFF, B. ; PRESLEY, A.: Structured models and dynamic systems analysis: the integration of the IDEF0/IDEF3 modeling methods and discrete event simulation. In: *WSC '97: Proceedings of the 29th conference on Winter simulation*. New York, NY, USA : ACM Press, 1997. – ISBN 0–7803–4278–X, S. 518–524 40

[Wid95] WIDOM, J.: Research Problems in Data Warehousing. In: *Proceedings of the 4th Int. Conf. Information and Knowledge Management (CIKM '95)*, 1995 32

[Wie02] WIENDAHL, H.-P.: Auf dem Weg zur „Digitalen Fabrik". In: *wt Werkstatttechnik online* 92 (2002), Nr. 4, S. 121 12

[Wik07a] WIKIPEDIA: *Computer simulation — Wikipedia, The Free Encyclopedia*. `http://en.wikipedia.org/w/index.php?title=Computer_simulation&oldid=105803227`. Version: 2007. – [Online; accessed 11-February-2007] 7

[Wik07b] WIKIPEDIA: *Java Architecture for XML Binding — Wikipedia, The Free Encyclopedia*. `http://en.wikipedia.org/w/index.php?title=Java_Architecture_for_XML_Binding&oldid=103161727`. Version: 2007. – [Online; accessed 19-February-2007] 111

[Wik07c] WIKIPEDIA: *Java Database Connectivity — Wikipedia, Die freie Enzyklopädie*. `http://de.wikipedia.org/w/index.php?title=Java_Database_Connectivity&oldid=26943789`. Version: 2007. – [Online; Stand 11. Februar 2007] 94

[Wik07d] WIKIPEDIA: *Java Naming and Directory Interface — Wikipedia, Die freie Enzyklopädie*. `http://de.wikipedia.org/w/index.php?title=Java_Naming_and_Directory_Interface&oldid=27002588`. Version: 2007. – [Online; Stand 19. Februar 2007] 95

[Wik07e] WIKIPEDIA: *Java (Programmiersprache) — Wikipedia, Die freie Enzyklopädie*. `http://de.wikipedia.org/w/index.php?title=Java_&oldid=27667739`. Version: 2007. – [Online; Stand 11. Februar 2007] 94

[Wik07f] WIKIPEDIA: *Model View Controller — Wikipedia, Die freie Enzyklopädie*. http://de.wikipedia.org/w/index.php?title=Model_View_Controller&oldid=27566045. Version: 2007. – [Online; Stand 11. Februar 2007] 55

[Wik07g] WIKIPEDIA: *SAP — Wikipedia, Die freie Enzyklopädie*. http://de.wikipedia.org/w/index.php?title=SAP&oldid=27602381. Version: 2007. – [Online; Stand 11. Februar 2007] 96

[Wik07h] WIKIPEDIA: *Service-oriented architecture — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Service-oriented_architecture&oldid=109151959. Version: 2007. – [Online; accessed 19-February-2007] 86

[Wik07i] WIKIPEDIA: XPath — Wikipedia, The Free Encyclopedia. (2007). http://en.wikipedia.org/w/index.php?title=XPath&oldid=105351868. – [Online; accessed 17-February-2007] 34

[Wil93] WILDEMANN, H.: *Optimierung von Entwicklungszeiten*. 1. München : Transfer-Centrum-Verl., 1993 43

[WKD00] WÖHE, G. ; KAISER, H. ; DÖRING, U.: *Einführung in die allgemeine Betriebswirtschaftslehre*. 9. München : Vahlens, 2000 v

[Wor02] WORTMANN, D.: Auf dem Weg zur „Traumfabrik“. In: *Fördern und Heben* 52 (2002), Nr. 3, S. 111f 12, 19

[ZB99] ZHANG, J. ; BROWNE, J.: New supporting tools for designing production systems. In: BRUSSEL, H. van (Hrsg.) ; KRUTH, J. P. (Hrsg.) ; LAUWERS, B. (Hrsg.): *Proceedings of the 32nd CIRP International Seminar on Manufacturing Systems*, 1999, S. 23–32 40

[ZFJ00] ZÜLCH, G. ; FISCHER, J. ; JONSSON, U.: An integrated object model for activity network based simulation. In: JOINES, J. A. (Hrsg.) ; BARTON, R. R. (Hrsg.) ; KANG, K. (Hrsg.) ; FISHWICK, P. A. (Hrsg.) ; IEEE (Veranst.): *Proceedings of the 2000 Winter Simulation Conference* Bd. 1. Piscataway, New Jersey, 2000, S. 371–380 xii, 77

[Zij00] ZIJM, W. H. M.: Towards intelligent manufacturing planning and control systems. In: *OR-Spektrum 22* 22 (2000), Nr. 3, S. 313–345 71, 72