Thomas Collisi

Ein informationslogistisches Architekturkonzept zur Akquisition simulationsrelevanter Daten

Thomas Collisi

Ein informationslogistisches Architekturkonzept zur Akquisition simulationsrelevanter Daten

Herausgegeben von
Professor Dr.-Ing. Klaus Feldmann,
Lehrstuhl für
Fertigungsautomatisierung und Produktionssystematik

FAPS

Meisenbach Verlag Bamberg

Als Dissertation genehmigt von der Technischen Fakultät
der Friedrich-Alexander-Universität Erlangen-Nürnberg

Tag der Einreichung:	15. Oktober 2001
Tag der Promotion:	28. Januar 2002
Dekan:	Prof. Dr. rer. nat. A. Winnacker
Berichterstatter:	Prof. Dr.-Ing. K. Feldmann
	Prof. Dr.-Ing. G. Reinhart, TU-München

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Collisi Thomas:
Ein informationslogistisches Architekturkonzept zur Akquisition simulationsrelevanter Daten /
Thomas Collisi. - Bamberg : Meisenbach, 2002
(Fertigungstechnik - Erlangen ; 117)
ISBN 3-87525-164-4

Herstellung: Gruner Druck GmbH, Erlangen-Eltersdorf
Printed in Germany

Vorwort

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Fertigungsautomatisierung und Produktionssystematik der Friedrich-Alexander-Universität Erlangen-Nürnberg.

Herrn Prof. Dr.-Ing. K. Feldmann, dem Leiter dieses Lehrstuhls am Institut für Maschinenbau und Fertigungstechnik, danke ich für die Förderung meiner Arbeit sowie den Freiraum zur Durchführung dieser Arbeit und für das mir entgegengebrachte Vertrauen.

Herrn Prof. Dr.-Ing. G. Reinhart, dem Leiter des Instituts für Werkzeugmaschinen und Betriebswissenschaften der Technischen Universität München, danke ich für die Übernahme des Korreferats.

Mein Dank gilt ferner allen Kollegen, Gutachtern und Projektpartnern im Bayerischen Forschungsverbund Simulationstechnik für eine sehr gute fachliche Zusammenarbeit. Speziell die Zusammenarbeit mit Dipl.-Ing. Johann Bayer und Dipl.-Ing. Matthias Hagmann von der BMW AG hat in einem überaus angenehmen Gesprächsklima viele Impulse für diese Arbeit gegeben.

Besonders herzlich danke ich Dipl.-Inf. Christoph Felbinger, Dipl.-Ing. Andreas Licha und Dr.-Ing. Thomas Stöckel für den jederzeit produktiven Gedankenaustausch und ihre Mitwirkung bei manchmal langwierigen Tests verteilter Applikationen. Ein genauso herzlicher Dank an Dipl.-Wirtsch.-Inf. Jürgen Wunderlich, mit dem ich zusammen einige interessante Projekte durchgeführt habe und der als Büronachbar stets für fachliche Diskussionen offen war.

Ferner gilt mein Dank allen Studenten, die durch Ihre Unterstützung die Arbeit in dieser Form ermöglicht haben. Im Besonderen möchte ich dabei den außergewöhnlichen Einsatz von Dipl.-Inf. Christos Hadoulas, Dipl.-Ing. Wilfried Kupfberger, Dipl.-Ing. Jürgen Hochfellner und cand. Ing. Daniel Craiovan hervorheben.

In privater Hinsicht gilt mein herzlichster Dank meiner Frau Simone, die mich nicht nur unterstützt und fortwährend motiviert hat, sondern deren Fachwissen und sorgfältige Korrekturen viel zum Gelingen dieser Arbeit beigetragen haben.

Erlangen, Januar 2002 Thomas Collisi

Ein informationslogistisches Architekturkonzept zur Akquisition simulationsrelevanter Daten

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Rapide und tiefgreifende Veränderungen der wirtschaftlichen Rahmenbedingungen, die Globalisierung von Unternehmen, technologische Entwicklungen sowie strategische Firmenentscheidungen vergrößern den Wettbewerbsdruck. Entsprechend wird die Wettbewerbsstärke der Unternehmen weitgehend von der Innovationskraft, Schnelligkeit und Flexibilität bestimmt [1].

Diese wettbewerbsbedingten Ansprüche werden technisch als höhere Anlagenauslastung, reduzierte Auftragsdurchlaufzeiten und Bestände sowie optimale Termintreue formuliert. Der Einsatz von E-Commerce fordert zudem die Produktindividualisierung, Verkürzung der Produktlebenszyklen, häufiges Neu- und Umplanen von Produktions- und Logistiksystemen und die Reduktion der Produkt- und Prozessentwicklungszeiten [2]. Entsprechend sieht sich der Planer in seinem ingenieurmäßigen Handeln einer steigenden Vielfalt von Einzelzielen gegenüber, die erfolgreich aufeinander abgestimmt werden müssen.

Eine unmittelbare Folge der Zunahme dieser Vielfalt ist ein drastischer Anstieg von Komplexität [3]. Um dieser funktions-, zeit- und kostengerecht zu begegnen, erfolgt die Planung meist verteilt und arbeitsteilig. Der Einsatz rechnergestützter Planungswerkzeuge, welche die Fachleute aus den unterschiedlichen Disziplinen wirkungsvoll beim Aufbau eines Problem- und Systemverständnisses unterstützen, ist unerlässlich. Der dadurch entstehende große Vernetzungszwang bei gleichzeitiger Zunahme der potentiellen Zielkonflikte zwischen den Experten wirkt sich ebenfalls komplexitätssteigernd aus.

Eine wichtige Rolle bei der Komplexitätsbeherrschung unter Verwendung umfassender Informationen nimmt die Simulationstechnik ein, deren Anwendungsfeld von der Planung bis zum Betrieb eines Produktionssystems reicht. Der besondere Vorteil dieses Werkzeugs ist, dass es weniger aufwendig ist als Umbauten oder der Prototypenbau. Hierdurch kann die unter Kostenkriterien wichtige Zeit zwischen Inbetriebnahme und Erreichen der Nennleistung einer Produktionsanlage (Anlaufphase) verkürzt werden. Vor allem die Tatsache, dass die Simulation den Zusammenhang zwischen Systemeigenschaften und Betriebsparametern anschaulich darstellt, trägt zur Akzeptanzsteigerung dieses Werkzeugs bei. Der messbare Mehrwert ist unbestritten [4].

Es zeigt sich jedoch, dass trotz immer weiter steigender Investitionen in leistungsfähigere Hard- und Softwaresysteme nicht zuletzt auch die Simulationstechnik von der mangelnden Durchgängigkeit der Informationstechnik beeinträchtig wird (Abb. 1): Informationen fehlen, sind inkompatibel oder können häufig nicht erreicht werden. Solchen 'Informationsinseln' folgen konsequenterweise dann Insellösungen, die den vorherrschenden Paradigmen nach Vernetzung und Globalisierung nicht gerecht werden können. Dies hat wiederum zur Folge, dass im Betriebsfall die bereitgestellten Ressourcen nicht exakt aufeinander abgestimmt werden. Die damit verbundene Ressourcenverschwendung schwächt letztlich die Wettbewerbsfähigkeit des Unternehmens. Die langfristige Gewinnmaximierung als Unternehmenszweck ist in Frage gestellt.

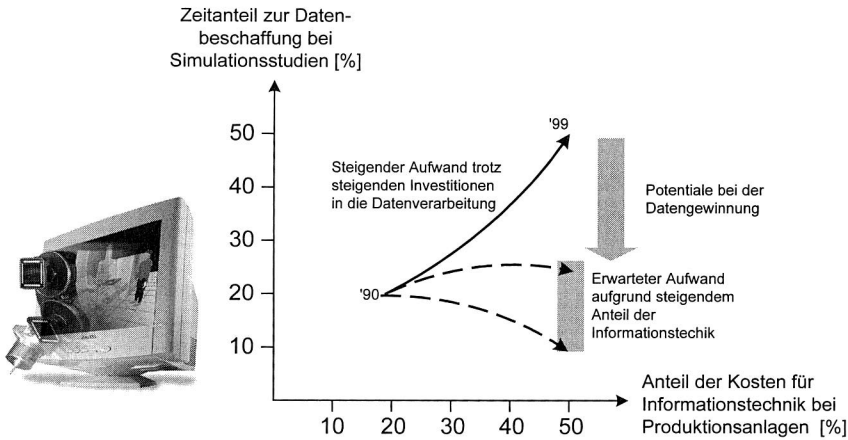


Abb. 1 *Gegenüberstellung des Investitionsanteils von Informationstechnik bei Produktionsanlagen und des Zeitanteils der Datenbeschaffung bei Simulationsstudien in den Jahren '90 und '99 [4,18,25,65]*

Ziel muss es sein, die einzelnen Planungswerkzeuge und Resultate der Experten durchgängig miteinander zu verbinden. Nur so lassen sich die Vorteile und Synergien, die durch den breiten Einsatz der einzelnen Planungstools erwartet werden, auch erreichen.

1.2 Ziele und Vorgehensweise

Unter dem Blickwinkel der Ablaufsimulation wird diese Arbeit die informationstechnischen Aspekte der durchgängigen Beschaffung und Verarbeitung simulationsrele-

vanter Informationen als häufig genanntes Anwendungshemmnisses aufgreifen. Es gilt, die Lücke zwischen den realen (datentechnischen) Gegebenheiten und den verfügbaren technischen Lösungen der Simulationstechnik zu schließen und den dadurch entstandenen Nutzen eines solchen Konzepts für Planer und Betreiber zu verdeutlichen. Das Ziel ist damit eine zielgerichtete Ergänzung der bereits bestehenden Systeme um ein informationslogistisches Architekturkonzept, mit dem die Beschaffung (Akquisition) simulationsrelevanter Daten für die Planung und den Betrieb weiter systematisiert und deutlich vereinfacht wird.

Die zur Entwicklung dieser Architektur gewählte Vorgehensweise beginnt zunächst damit, den System- und Modellbegriff bei Produktionssystemen näher zu untersuchen. Dieser leitet direkt zur Simulation über, die als Bewertungsmethode der entsprechenden Systembeschreibungen herangezogen wird. Die Betrachtung der einzelnen Phasen einer Simulationsstudie führt dann zu den damit verbundenen Defiziten. Gleichzeitig wird aber auch verdeutlicht, dass besonders die häufig genannte Komplexität von Simulationsmodellen und der dafür notwendigen Datenmenge nicht künstlich ist, sondern vielmehr die Komplexität der ursprünglichen und zukünftigen Systeme widerspiegelt. Die Erkenntnis, dass in anderen aktuell diskutierten Bereichen mit Bezug zur Informationstechnik, etwa des eCommerce oder der Webbasierten Anwendungen, bereits eine wesentlich höhere Anzahl universellerer Datenaustauschmöglichkeiten vorhanden ist, motiviert, Ähnliches für die Simulation zu schaffen. Erster Schritt dazu ist eine umfassende Analyse simulationsrelevanter Daten. Dies ergibt zuletzt eine kompakte Beschreibung von Datenklassen, die den Informationsbedarf während des Entwurfs eines Simulationsmodells sowie während der Vorbereitung und Durchführung von Simulationsexperimenten beschreiben.

Die Übersicht des Informationsgehalts der Datenklassen bildet im Anschluss die Grundlage, um die Anforderungen an eine Datenkopplung zwischen der (geplanten) Anlage und der Simulation zu analysieren. Diese werden unter der Maßgabe zweier allgemeiner Szenarien vorgenommen, die sowohl die planungsbegleitende als auch die betriebsbegleitende Anwendung der Simulation abdecken. Die darauf aufgebauten Anforderungen an ein System zur Verwaltung der Datenklassen leitet sich aus diesen in der Praxis gängigen Szenarien ab. Die Verbindung des Gesamtsystems erfolgt unter Berücksichtigung des Standes der Technik durch eine marktgängige Middleware. Dieses wird insgesamt zu einem Framework für Simulationsanwendungen ausgebaut.

Die wichtigste Stärke des Frameworks wird die konzeptionelle Flexibilität beim Einsatz verschiedener Simulationssysteme und der Nutzung unterschiedlichster Datenquellen sein. Den unternehmensstrategischen Begriffen 'Vernetzung' und 'Globalisie-

rung' wird dabei informationstechnisch durch Konzepte wie 'Ortstransparenz', 'Plattforminvarianz' und 'mehrschichtige Client/Server-Modelle' begegnet.

Im weiteren Verlauf der Arbeit wird das entwickelte System als Plattform für die Datenkopplung verwendet. Darauf aufbauend ist anschließend zu überlegen, wie das nun verfügbare zeit- und planungsnahe Datenmaterial genutzt werden kann. Dazu werden Werkzeuge entwickelt, die die Beobachtung der bereitgestellten Informationen gestatten, um ggf. unerwünschte Abweichungen (Störungen) sicher erkennen und verarbeiten zu können. Dieses Ergebnis führt dann zu einem simulationsbasierten online-Monitoring. Hier werden die Ergebnisse der Datenkopplung und -überwachung in einer Anwendung zusammengeführt, bei der ein Simulationsmodell permanent aus dem Betriebsgeschehen aktualisiert wird. Das damit entstehende Abbild des beobachteten Systems wird ständig auf signifikante Abweichungen überwacht. Bedarfsweise werden dann Alarme ausgelöst, die eine simulative Bewertung der vorliegenden Situation anstoßen, in einer Datenbank hinterlegte Gegenmaßnahmen bewerten und dem Anwender ein angemessenes Handeln ermöglichen.

Zuletzt werden die vorgestellten Ergebnisse unter dem Gesichtspunkt der Integration in die betrieblichen Abläufe diskutiert. Es wird sich ein dreistufiger Ansatz zur Implementation eines solchen Frameworks ergeben, welcher sich an dem häufig vorzufindenden informationstechnischen Umfeld von Produktionssystemen bei Planung und Betrieb orientiert und die Stärken des entwickelten Konzept nochmals deutlich hervorhebt.

2 Modellierung und Bewertung von Produktionssystemen

Ein Produkt durchläuft bei seiner Herstellung eine Folge von Arbeitsstationen, deren Anordnung durch die Struktur des Produktionssystems festgelegt wird. Aufgabe der Arbeitsstationen ist es, durch Anwendung verschiedener Verfahren (Urformen, Umformen, Beschichten, Trennen, Fügen oder Ändern der Stoffeigenschaften [5]) genau die Funktionen auszuführen, die im einzelnen durch den Produktionsplan für das jeweilige Produkt vorgesehen sind. Unterschieden werden kann hierbei ferner zwischen wertschöpfenden und nicht wertschöpfenden Stationen - die Trennung erfolgt dabei anhand des messbaren Wertzuwachses im Sinne der Funktionserfüllung. Arbeitsstationen sind Maschinen, manuelle Arbeitsplätze oder automatische Anlagen, welche die hierzu notwendige zielgerichtete Verknüpfung/Veränderung von Roh- und Hilfsstoffen, sowie Informationen und Energie zur Erfüllung dieser Aufgaben erbringen.

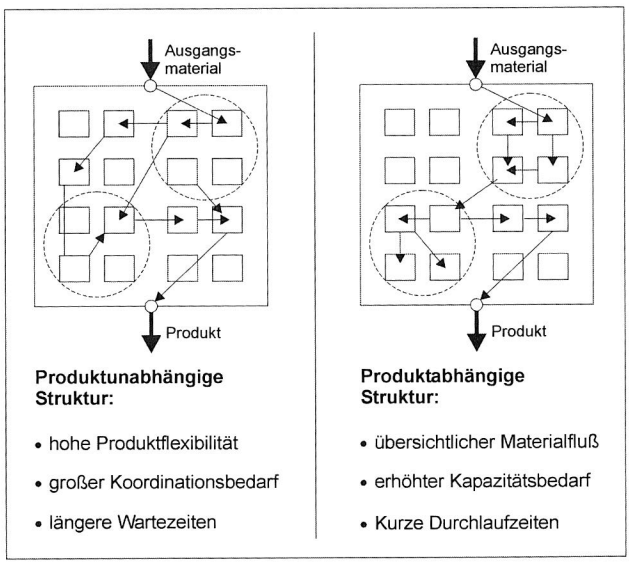


Abb. 2 Zusammenhang zwischen Produktionsaufgabe und Produktionsstruktur

Hinsichtlich der Produktions*aufgabe* besteht dabei ein enger Zusammenhang mit der Produktions*struktur*. Hierdurch wird maßgeblich bestimmt, inwieweit die resultierende Arbeitsablauffolge eines Produkts technisch und wirtschaftlich sinnvoll ist. Es ist jeweils ein Kompromiss zwischen produktunabhängigen und produktabhängigen Strukturen zu finden, bei der wichtige Faktoren wie Rüstaufwand, Automatisierungs-

grad, Flexibilität und Ausrichtung der Betriebsmittel am Materialfluss aufeinander abgestimmt werden müssen (Abb. 2).

Da ein solcher Kompromiss technische und wirtschaftliche Randbedingungen impliziert, die gleichzeitig die Ausführung von Aufträgen einschränken, muss neben der Struktur des Produktionssystems parallel eine Organisation vorhanden sein, die die verbleibenden Freiheitsgrade durch optimale *Abläufe* ausschöpft. Typische Aufgaben der Organisation sind die Bereitstellung von Logistikfunktionen (Lagerung, Bewegung, Verteilung), die Verwaltung von knappen Betriebsmitteln und die Interaktion mit Zulieferern und Abnehmern (Märkten) [6,7,8].

Diese Organisation ist besonders bei modernen Produktionsunternehmen gefordert, da diese häufig nur noch schwer überschaubare Wirkungsgefüge darstellen. Speziell bei Unternehmen mit einem breiten und tiefen Erzeugnisspektrum, unterschiedlichen Produktionstypen und verteilten Standorten wird es immer zwingender, Ansätze zu wählen, mit denen die hohe Anzahl an Betrachtungsobjekten und deren - meist sogar gegenläufigen oder widersprüchlichen - Beziehungen untereinander beherrschbar bleibt [7]. So muss beispielsweise in der strategischen Unternehmensplanung das Zusammenspiel von Werken und Lieferanten bekannt sein, um mögliche Auswirkungen einer Entscheidung zu berücksichtigen. Planer und Entscheider besitzen zwar das umfassende Detailwissen über Produktionsstruktur, Artikelsortiment und Struktur des Planungs- und Steuerungssystems des Unternehmens, dennoch ist die Vielzahl der zu berücksichtigenden Einflussgrößen keinesfalls trivial. Bei steigender Komplexität wächst damit, bedingt durch ein lückenhaftes oder falsches Problem-, System- oder Parameterverständnis, auch das Risiko einer Fehlentscheidung.

2.1 Der Modellbegriff bei Produktionssystemen

Zur Beherrschung der zunehmenden Komplexität von Produktionssystemen werden Methoden benötigt, die die Chance bieten, auf einem handhabbaren Komplexitätsniveau die erfolgreiche Weiterentwicklung des Unternehmens voranzutreiben [9]. Die eigentliche Methodik ist dabei vielfach von einem Systemaspekt geprägt [10]. Hierbei wird das Produktions*system* als eine Menge von Elementen betrachtet, die zueinander in Relation stehen und mit ihrer Umgebung verbunden sind. Der meist genutzte Ansatz ist problem- oder zweckorientierter und betrachtet nur Elemente, die in bestimmter, d.h. von außen beobachtbarer oder beschreibbarer Weise aufeinander einwirken. Weiterhin ist es jederzeit möglich, Hierarchien aus Systemen und Teilsystemen zu bilden, so dass die Beschreibungsgenauigkeit eines Systems im Rahmen eines Detaillierungsprozesses iterativ dem Problem- und Systemverständnis

folgen kann [11,12]. Für ein Produktionssystem werden hieraus beispielsweise die Ebenen Fabrik, Anlage, Zelle und Komponente schrittweise abgeleitet (Abb. 3). Es ergibt sich also ein modularer und hierarchischer Systemaufbau, in dem verschiedene Betrachtungsinhalte (z.B. Fertigungsprinzip, Anlagenlayout, Takzeiten, NC-Programme usw.) vereint werden können [13].

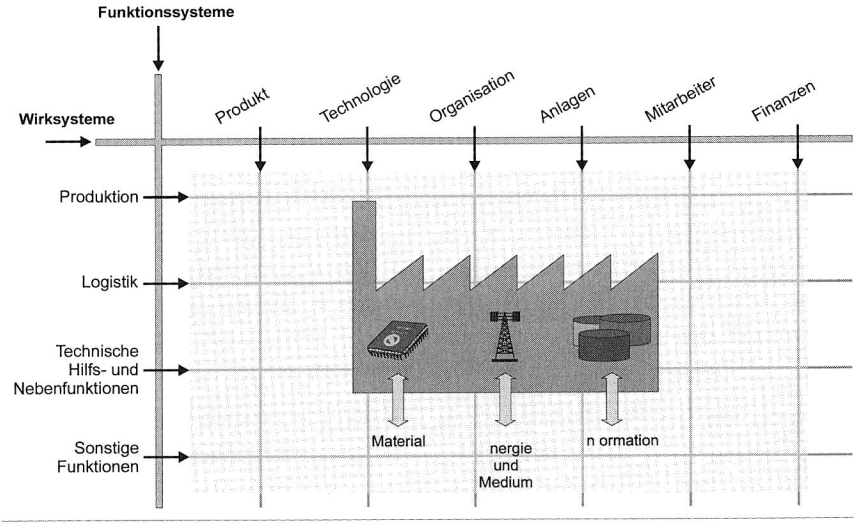


Abb. 3 Sichten und Betrachtungsinhalte des Systems "Fabrik"

Die systemtechnische Vorgehensweise ist dabei durch einen schrittweisen Prozess gekennzeichnet, bei dem die gestellte Aufgabe zunächst systematisiert und formal dargestellt wird. Systematisierung ist notwendig, um mit geringem Zeitaufwand eine angemessene Lösung zu erreichen [14], mit der Entwicklungs- und Projektierungsprobleme präzisiert und objektiviert werden können. Formalisierung erschließt die Einsatzmöglichkeiten der elektronischen Datenverarbeitung, der Kommunikation und damit einer interdisziplinären Arbeitsweise [15]. Ziel ist hierbei, eine "gemeinsame Sprache" zwischen den Beteiligten zu finden, was in den meisten Fällen dadurch gelingt, dass die zu betrachtenden Strukturen und Abläufe modellmäßig beschrieben und anschließend die Fachsichten der einzelnen Beteiligten vereinigt werden. Somit hängt es maßgeblich von den Untersuchungszielen ab, was unter den Begriffen 'System' und 'Modell' zu verstehen ist [16]. Besonders bei der Planung und dem Betrieb von Produktionssystemen sind meist mehrere parallele Sichten (=Modelle) notwendig, um der Komplexität gerecht zu werden. Zu nennen seien hier die Organisations-, Produkt-, Prozess-, Steuerungs-, Ressourcen-, Funktions- und Datensichten.

Mit solchen Modellen kann der Sachverhalt dann vereinfachter und übersichtlicher dargestellt werden, so dass sich die notwendigen Erkenntnisse hieraus leichter ableiten lassen. Ein Modell kann somit als ein "gedanklich isolierter Teilzusammenhang" verstanden werden, mit dem innerhalb "eines vom Untersuchungsziel abhängigen Toleranzrahmens" Erkenntnisse über die Grundzusammenhänge des Ausgangssystems gewonnen werden können [17,18,19].

2.2 Simulation

Zwar trägt bereits der Modellaufbau zu einem Systemverständnis bei, jedoch reicht dieser Schritt meist nicht aus, da Produktionssysteme definitionsgemäß dynamisch sind. Die enthaltenen Systeme, Teilsysteme und Elemente weisen ein zeit- und zustandsabhängiges Verhalten auf, welches von deterministischen und/oder stochastischen Einflüssen überlagert wird. Solange die Ausgangssysteme überschaubar sind, können möglicherweise noch Experimente am realen System durchgeführt werden oder mathematische Verfahren liefern analytische Lösungen (z.B. Warteschlagen, Wahrscheinlichkeitstheorie oder Petri-Netze [20]).

Bei hinreichend komplexen Systembetrachtungen wird es immer wichtiger, sowohl den Modellaufbau als auch die Analyse und Bewertung der Modelle mit rechnergestützten Werkzeugen durchzuführen. Hierbei zeigt sich, dass das Werkzeug 'Simulation' eine zunehmende Verbreitung findet, da die Komplexität im produktionstechnischen Umfeld mit seinen verschiedenen Fachsichten sehr gut in einem Simulationsmodell abgebildet werden kann. Unter Berücksichtigung der vielfachen dynamischen Wirkzusammenhänge ist hiermit eine umfassende Bewertung von Produktionssystemen möglich. Unter Simulation wird "*das Nachbilden eines Systems mit seinen dynamischen Prozessen in einem experimentierfähigen Modell*" verstanden, "*um zu Erkenntnissen zu gelangen, die auf die Wirklichkeit übertragbar sind*" [18].

Mit der Simulation findet sich ein Werkzeug, welches in der betrieblichen Praxis anerkannt ist und mit dem ein großer Anwenderkreis zum überwiegenden Teil bereits positive Erfahrungen macht [21,22]. So zeigt sich, dass durch den Einsatz der Simulation in Produktion und Logistik eine verbesserte Planung bei gleichzeitiger Kostensenkung zu erwarten ist [23]. Ebenso bietet die Simulation die Möglichkeit, neue oder "unübliche" Strukturen und Abläufe gut bewerten zu können, wie am Beispiel einer Gegenüberstellung zweier Fertigungsmethoden (Komplettbearbeitung, konventionelle Einzelmaschine) in [24] gezeigt wird. Eine aktuelle Einschätzung des Nutzens ist in Abb. 4 zu finden. Eine hiervon unabhängig durchgeführte Umfrage bestätigte, dass Simulation ein von Planern und Entscheidern anerkanntes Werkzeug ist [25],

welches bei der Bewertung und Optimierung von Produktionssystemen erfolgreich eingesetzt wird.

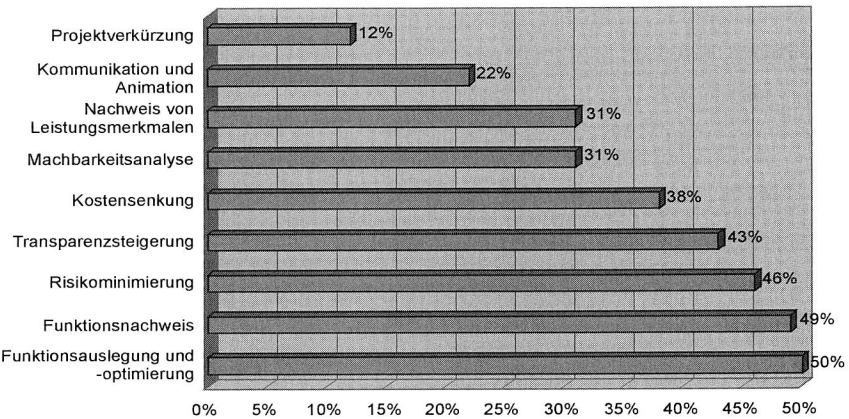


Abb. 4 Typische Anwendungsfelder der Simulation (Mehrfachnennungen möglich)

2.3 Durchführung von Simulationsstudien

Eine Simulationsstudie besteht aus mehreren verschiedenen Phasen, die im folgenden erläutert werden. Unter Umständen werden einzelne Phasen hierbei mehrfach durchlaufen. Diese Zusammenhänge sind in Abb. 5 veranschaulicht. Unabhängig von der Problematik und dem gewählten Abstraktionsgrad ist die Vorgehensweise grundsätzlich immer gleich.

Zunächst muss für eine Simulationsstudie eine möglichst präzise Formulierung des Problems und des angestrebten Ziels erfolgen. Die damit verbundenen Vorgaben dienen dann in der nächsten Phase dazu, eine allgemeine Strukturanalyse durchzuführen.

Die Strukturanalyse klärt grobe Zusammenhänge innerhalb des Problemfeldes ab. Ihre Ergebnisse sind Grundlage der Modellerstellung. Beispielsweise werden dabei vorhandene Hierarchien und funktionale Einheiten identifiziert, um diese berücksichtigen zu können. In diesem Zusammenhang bedeutet dies, alle benötigen Daten zu dem betreffenden Unternehmen als Strukturinformationen in einer Datenbank zu hinterlegen, so dass abhängig von der Zielstellung die passenden Daten schnell extrahiert werden können.

Nachdem die Struktur des Problems bekannt ist, muss in einem weiteren Schritt für diese Struktur eine Datendefinition und eine Datenerhebung stattfinden. Ziel ist es, für alle Elemente der Struktur eine ausreichend genaue Beschreibung zu finden. Für die Simulation von Produktionsanlagen sind dies beispielsweise Zeitdaten, die das dynamische Verhalten beschreiben wie etwa Transport-, Stör- und Bearbeitungszeiten.

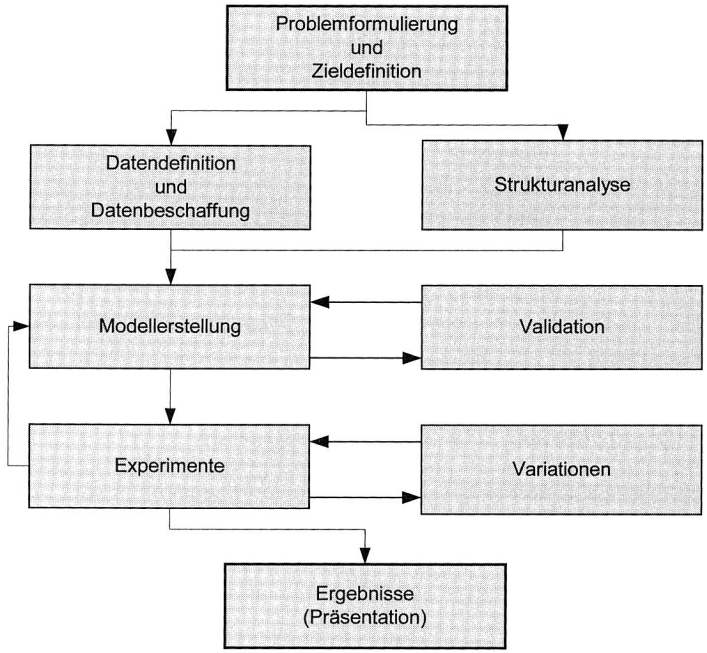


Abb. 5 Ablauf einer Simulationsstudie

Aus der vorhandenen Struktur und den erhobenen Daten kann nun in der Phase der Modellerstellung ein Simulationsmodell aufgebaut werden. Alle gesammelten Erkenntnisse des Verhaltens fließen unter Berücksichtigung der Ziele und Vorgaben hier ein.

An die Modellerstellung schließt sich eine Validierung des Simulationsmodells an. In diesem Schritt werden die Zuverlässigkeit und Gültigkeit des Modells anhand von ausgewählten Daten und ersten Simulationsexperimenten geprüft. Dieser Schritt ist besonders wichtig, da die Qualität und das Vertrauen in die Ergebnisse entscheidend davon abhängen, dass zunächst bekannte Sachverhalte im Simulationsmodell korrekt wiedergegeben werden. Ohne dieses Vertrauen sind spätere Experimente zwar

durchführbar, jedoch fehlen in diesem Fall die Argumente, um einer späteren kritischen Überprüfung standzuhalten [26].

Hat das Simulationsmodell in der Validierungsphase überzeugt, wird mit Simulationsläufen, Analysen und Optimierungen begonnen. Ab diesem Punkt werden die anfänglichen Fragen beantwortet und es können Optimierungskreisläufe angestoßen werden. Weiterhin ist es möglich, Analysen zu Fragen zu erstellen, die auf einfachen, z.T. spontanen Ideen beruhen ("Was wäre, wenn ...?").

Abschließend erfolgt die Präsentation der Simulationsergebnisse. Die Präsentation ist in gewisser Weise die Schnittstelle zu den Entscheidungsträgern, die die zukünftigen Schritte eines Unternehmens auch auf die Ergebnisse der Simulation stützen.

2.4 Defizite beim Simulationseinsatz

Obwohl die Simulation zahlreiche Stärken aufweist, bestehen eine Reihe von Gründen, die den systematischen Einsatz der Simulation in der Praxis erschweren. Die Argumente hierzu können in folgende Bereiche eingegliedert werden (Abb. 6):

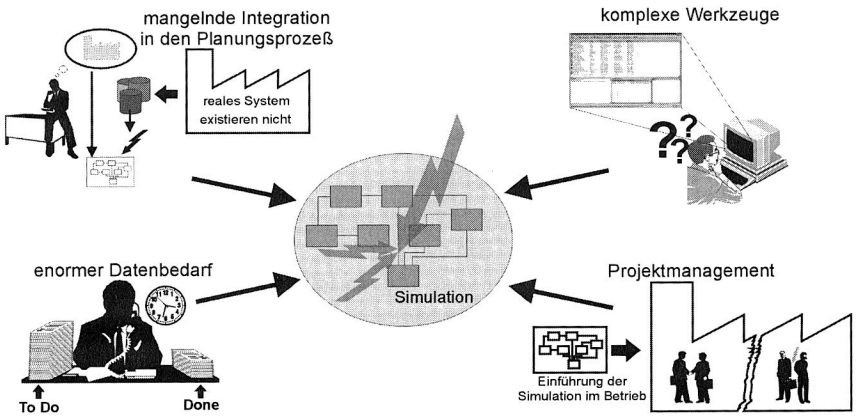


Abb. 6 Defizite beim Simulationseinsatz

- Hohe Komplexität der Werkzeuge
- Projektmanagement, -durchführung und Unternehmensumfeld
- Mangelnde Integration in den Planungsprozess
- Enormer Datenbedarf

2.4.1 Komplexität der Werkzeuge

Hinsichtlich der Komplexität kritisieren viele Anwender die verfügbaren Werkzeuge und deren aufwendige Bedienung. Dies ist einerseits darauf zurückzuführen, dass für die Simulation zwar immer weniger Spezialkenntnisse verlangt werden, jedoch anwenderseitig für die Modellbildung eine hohe Abstraktionsfähigkeit vorhanden sein muss. Es handelt sich um keine künstliche Komplexität, die nicht vorhanden wäre, wenn nicht simuliert werden würde. Vielmehr ist die Komplexität des Simulationsmodells ein Ausdruck einer realen Komplexität [27]. Dieser müssen folglich die Methoden (durch Vorgehensweisen) und Anwender (durch Fähigkeiten), aber auch die Werkzeuge (durch Möglichkeiten) folgen. Darüber hinaus ist deutlich zu erkennen, dass ganz besonders der letzte Punkt von den Herstellern von Simulationssystemen aufgegriffen wird, da Akzeptanz, d.h. Markterfolg, maßgeblich von der Anwenderfreundlichkeit abhängt [22].

2.4.2 Projektmanagement, -durchführung und Unternehmensumfeld

Der Punkt Unternehmensumfeld, Projektmanagement und -durchführung ist eng mit dem Ansehen der Simulation bei den Entscheidungsträgern und Anwendern verbunden. Damit ist gemeint, dass die Simulation bei allen Beteiligten genutzt wird und anerkannt ist. Dabei wird von Unternehmen bisweilen angemerkt, dass das Aufwand-/Nutzenverhältnis ungünstig sei. Es wird jedoch festgestellt, dass die Ursache für diese Haltung wesentlich in einer Bewertung gesehen wird, die die weniger quantifizierbaren Leistung der Simulation außer Acht lassen [28], beispielsweise die Notwendigkeit zur abteilungsübergreifenden Abstimmung von Sichtweisen und Begrifflichkeiten. Für eine erfolgreiche Integration der Simulation in das betriebliche Umfeld ist daher entscheidend, dass der Simulationseinsatz im Unternehmen durch eine strategische Entscheidung gefestigt wird, wie beispielsweise Berichte aus großen Konzernen zeigen [29]. In diesem Schritt wird dann auch die Möglichkeit erkannt, ein Projektmanagement zu etablieren, welches die Potentiale der Simulation zielgerichtet und qualifiziert in der Projektarbeit umsetzen kann. Das Projektmanagement muss auch dabei unterstützt werden, die Komplexität des Planungsobjekts auf verschiedenen Arbeitsgruppen zu verteilen [30], ohne dass dabei ein Kommunikationsproblem - als Ausdruck unterschiedlicher Sichtweisen - an der Schnittstelle zwischen den einzelnen Gruppen den Projekterfolg gefährdet.

2.4.3 Mangelnde Integration in den Planungsprozess

Mit dem Problem der Projektdurchführung ist häufig auch eine mangelnde Integration in den Planungsprozess [31] und ein Nebeneinander verschiedener Anwendungen verbunden. Unter Integration wird das Ziel verstanden, durch Vereinigung, Verbin-

dung oder Vereinheitlichung eine "Ganzheit" zu erreichen. In den meisten Fällen scheitert ein Integrationsversuch an einer mangelnden Einheitlichkeit der unterschiedlichen Werkzeuge, etwa der Simulation und der Layoutplanung. Dies setzt sich in der Weiterverwendung der Simulationsergebnisse fort, sodass die mangelnde Integration in den Planungsprozess sich häufig darin manifestiert, dass einzelne Elemente entlang der Planungskette nicht in direkter funktionaler Verbindung zueinander stehen oder Medienbrüche vorliegen [32]. Dies hemmt besonders dann den effizienten und nachhaltigen Simulationseinsatz, wenn ein Simulationsmodell innerhalb einer Planung mehrfach eingesetzt werden soll. In diesem Fall kumulieren sich die Reibungsverluste an den Systemgrenzen der Werkzeuge im Projektverlauf und der wirtschaftliche Einsatz jedes einzelnen Werkzeugs entlang der Planungskette wird in Frage gestellt. Ein solcher Fall liegt beispielsweise in dem oft als problematisch bezeichneten Zusammenspiel zwischen CAD- und Simulationswerkzeugen vor [33], bei dem ein Medienbruch zwischen den Werkzeugen und entlang der Fachaufgaben 'Planung' und 'Simulation' verläuft. Aktuelle Berichte zeigen speziell im Bereich der Kopplung von CAD- und Simulationssystemen vielversprechende Aktivitäten der marktführenden Hersteller auf. Gleichzeitig weisen die gleichen Berichte jedoch darauf hin, dass die notwendigen Grundlagen zur Anbindung der Ablaufsimulation in die Prozessplanung, das Projektmanagement und die ganzheitliche Produktionsoptimierung noch erhebliche Aufwendungen verlangen [34,35,36]. Im Zusammenhang mit dem Trend zur Digitalen Fabrik werden demzufolge auch eher Chancen also Potentiale aufgezeigt, wie beispielsweise [37] verdeutlicht.

Entsprechend wird durch die Defizite auch die Arbeit des Projektmanagements in der Durchführung des Projekts belastet, d.h. für das Projekt benötigte Ressourcen können frühzeitig kaum eingeplant und Termine häufig nicht eingehalten werden. Aus der Sicht der Unternehmensführung bzw. der Auftraggeber entsteht hierdurch der Eindruck, dass die Beteiligten unzureichend qualifiziert sind, das Projekt unzureichend vorbereitet haben, Einflussgrößen unvollständig beachtet haben, über eine mangelhafte Kenntnis der Randbedingungen verfügen, die geforderten Experimente fehlerhaft durchgeführt haben und/oder die Ergebnisse falsch interpretieren [38, 39]. Dieses Defizit wird die vorliegende Arbeit aufgreifen.

2.4.4 Enormer Datenbedarf

Wird die Problematik des Informationsflusses entlang der Planungskette aus Sicht der Simulation genauer untersucht, ist zunächst zwischen dem Informationsbedarf und der Informationsverfügbarkeit zu unterscheiden, was bei der Diskussion dieser Problematik häufig vermischt oder synonym verwendet wird.

Ausgangspunkt ist der von vielen Quellen genannte enorme Datenbedarf, den die Entscheidung zur Durchführung einer Simulationsstudie nach sich zieht [40,41,42]. Ähnlich der Modellkomplexität ist der Datenbedarf ein Ausdruck für den Anspruch an ein qualitativ hochwertiges Abbild des realen oder geplanten Systems (im Sinne einer Charakterisierung). Deshalb ist auch hier die Simulation nicht der Auslöser für den Datenbedarf, sondern spiegelt vielmehr den erhöhten Anspruch an die Genauigkeit der Aussagen wider, die unter Verwendung des Werkzeugs Simulation durch Experimente an einem Simulationsmodell gewonnen werden sollen. Quantitativ kann dies darin gesehen werden, dass die Simulation zeit- und zustandabhängige Informationen fordert, die klassische statische Methoden nicht verarbeiten können, daher auch nicht benötigen und folglich auch keine Primärdatenquelle verfügbar ist [43]. Besonders das zeitliche Verhalten aller betrachteten Elemente - einzeln und in ihrer Gesamtheit - kann bei statischen Verfahren meist außer Acht gelassen werden und deshalb ist in diesem Fall ist der Datenbedarf entsprechend geringer. Daher nennen Quellen einen Zeitanteil von bis über 50% für die Datenerhebung an einer Gesamtstudie [21,25,28], wozu eine differenzierte Untersuchung anhand der Vorgehensweise bei Simulationsstudien im folgenden Kapitel vorgenommen wird.

2.5 Eingesetzte Verfahren zur Datenbeschaffung

Zur Reduzierung des Aufwands der Datenbeschaffung wird typischerweise der Ansatz vorgeschlagen, bereits vorhandene Unternehmensdaten zu nutzen [44]. Allerdings stoßen die für die Simulation Verantwortlichen und Ausführenden häufig an den Punkt, dass die vorhandenen Informations- und Organisationsstrukturen nicht kompatibel sind, dass die Informationen nicht in einer einheitlichen oder vereinheitlichten Form vorliegen oder hinsichtlich der Qualität nicht den Erfordernissen für das Simulationsprojekt entsprechen. Hierbei ist jedoch zu bemerken, dass besonders große Unternehmen in vielen anderen Bereichen durchaus über ein effektives Datenmanagement für Planung und Betrieb von Produktionssystemen verfügen. Dennoch sind die verfügbaren Systeme nicht auf die Deckung des Primärdatenbedarfs für Simulationszwecke abgestellt. Im weiteren Verlauf dieser Arbeit werden daher auch Überlegungen angestellt, inwieweit die notwendigen Informationen tatsächlich nicht vorhanden oder lediglich nicht erreichbar sind (Abb. 7) [45,46,47,48,49].

Die Ausgangslage stellt sich jedoch so dar, dass die vorhandenen Informationsmengen lückenhaft und die Beseitigung des vorgehend genannten Mangels nur durch einen erheblichen Einsatz an Personal und Zeit erfolgen kann. Vom Standpunkt der Simulation aus fehlt also beispielsweise ein Dokumentationssystem, welches im Sinne eines Stücklistenverfahrens die Zusammensetzung und Verwendung von Daten-

komponenten in Informationssystemen für Simulationszwecke strukturiert. Die dazu notwendigen Schemata zum Aufbau und zur Nutzung der jeweiligen Dokumentenstruktur sind zwar konzeptionell beschrieben (z.B. bei [50]), jedoch bei Unternehmen höchstens punktuell im Einsatz [51]. Somit fehlen im Bereich der Simulation noch Systeme, die die für die Simulation notwendigen Metadaten (Daten über Daten, Datenquellen, Programme usw.) sowohl für technische Systeme, also auch für Benutzer, Entwickler und Manager zur Verfügung stellen [52].

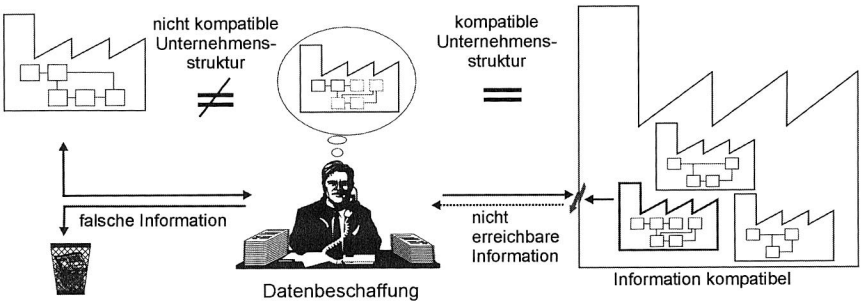


Abb. 7 Datenbeschaffungsproblematik aufgrund nicht vorhandener, inkompatibler oder nicht erreichbarer Informationen

Im Gegensatz zum Einsatz der Simulation und deren Integration in die betrieblichen Abläufe haben sich speziell im Zusammenhang mit der Verbreitung der Internettechnologien als Massenkommunikationsmittel im kommerziellen Bereich Systeme entwickelt, die die Schnittstellen zwischen Firmen (Business-to-Business) oder zum Endkunden (Business-to-Consumer) neu beschreiben und tiefgreifend verändert haben. Diese Schnittstellen bilden die Plattformen für "Strategische Allianzen", "Outsourcing" und "virtuelle Unternehmen". In diesem Zusammenhang geht Deloitte Research davon aus, dass sich auf der Basis genau strukturierter Informationen bis 2002 zwischen Unternehmen ein Umsatzanteil von rund 80% im Internet entwickeln wird [53].

Aus der Erkenntnis des zentralen Wertes von Informationen im Wettbewerb hat sich der Gedanke des Wissensmanagements entwickelt. Ziel ist es, im Unternehmen vorhandene Informationen und Wissen zu erschließen und als wertschöpfende Ressourcen zu nutzen. Entsprechend wird in einer Ovum Studie bis 2002 für Wissensmanagement ein jährliches Wachstum von 54% im Bereich Software und 35% im Bereich Dienstleitungen prognostiziert [54].

Zur Implementierung des Wissensmanagements kommen vorzugsweise Internettechnologien zum Einsatz, für die sich mit großer Geschwindigkeit derzeit ein eigener Markt entwickelt. Dieser Trend entspricht gleichfalls den Diskussionen vieler Ent-

wickler und Nutzer von Simulationswerkzeugen. Arbeiten zum Thema "Web-Based-Simulation" sind eine Facette dieser Bestrebungen [55, 56, 57]. Es wird überlegt, wie unter Verwendung des world-wide-web Simulationsmodelle

- entworfen,
- dokumentiert,
- analysiert und
- ausgeführt werden können.

Es wurde erkannt, dass Web-Based-Simulation analog zu den genannten eCommerce-Bereichen einen Paradigmenwechsel (im Sinne von [58]) nach sich zieht und die bekannten Modellierungstechniken radikal verändern wird - Simulationsmodelle werden stärker als in sich konsistenter Informationsblock betrachtet, der Wissen konserviert und demzufolge integraler Bestandteil eines Wissensmanagementsystems werden kann. Die Beschreibungsmethoden des WWW unterstützen dabei in einfacher Weise dieses Vorgehen durch Navigation, natürliche Hierarchien und Plattformunabhängigkeit. Aufgaben sind damit leichter zu strukturieren und zu organisieren.

Trotz dieser aussichtsreichen Bestrebungen ist in allen Fällen (eCommerce, Web-Based-Simulation, Wissensmanagement) festzuhalten, dass hier keine allgemein anerkannten Konzepte verfügbar sind, welche die Lücke zwischen den verfügbaren Techniken und den notwendigen Anwendungen schließen. Was im Rahmen von e-Commerce durch ein immenses Marktpotential mittelfristig gelingen wird, ist für die Simulation noch nicht abzusehen. Bei den diskutierten Anwendungen handelt es sich entweder um Prototypen, deren produktiver Einsatz noch aussteht oder um Sonderapplikationen, die indirekt bereits hohe Anforderungen an das betriebliche Umfeld stellen. Diese Anforderungen beziehen sich auf Systeme, deren Funktionsfähigkeit umfangreiche Voraussetzungen erfüllen muss, um letztlich einen im Verhältnis zur Investition geringen Simulationsmehrwert zu erreichen.

Festzuhalten bleibt insgesamt, dass Informationsquellen nicht konsequent genutzt werden [59]. Besonders in hochautomatisierten Produktionssystemen, bei denen die Simulation (hier konkret: die Ablaufsimulation) ein erprobtes Werkzeug während der Planung und des Betriebs solcher Systeme ist, können die vorhandenen Informationen nicht direkt bis zum Simulationsmodell geleitet werden. Dementsprechend stellt eine Datenkopplung zwischen solchen Anlagen und der Simulation derzeit für die Betreiber einen doppelten Aufwand dar, weil sowohl die Informationsinfrastruktur für reine Produktivapplikationen aufgebaut als auch die Informationsbereitstellung an die Bedürfnisse der Simulation mit hohem Aufwand angepasst werden müssten.

Herkömmliche Ansätze zur Nutzung der Informationsbestände für die Simulation zielen zumeist auf eine explizite Kopplung an ein spezielles Werkzeug ab, z.B. ein PPS–System, bzw. auf die Kopplung an eine Datenbank über eine Datenbankschnittstelle [60, 61]. Die Vorteile dieser Vorgehensweise bestehen dabei in der Einbindung vorhandener Werkzeuge und Datenbestände. Jedoch müssen die relevanten Informationen einerseits genau vorstrukturiert werden. Andererseits ist der Austausch von Daten mit einem PPS–System oder einer Datenbank mit einer hohen Latenzzeit verbunden, da solche Systeme üblicherweise nicht für die Deckung des Primärdatenbedarfs von Simulationsstudien konzipiert sind. Insbesondere bei der Durchführung von Projekten mit dem Ziel der Optimierung und Restrukturierung bestehender Anlagen, bei denen also ein direkter dynamischer Bezug zur abzubildenden Realität herzustellen ist (online-Kopplung), steht jedoch die permanente Aktualität der Daten stark im Vordergrund. Gleichzeit sollen/können bestehende Datenverarbeitungssysteme häufig nicht mit der zur Simulation notwendigen Datenbeschaffung und -aufbereitung belastet werden, oder es zeigt sich, dass viele Datenbanksysteme einer solchen Auftragslast nicht gewachsen wären [62]. Daraus ergibt sich dementsprechend aus Sicht der Simulation eine unbefriedigende "*Quality of Service*" für solche Datenverarbeitungssysteme: Je näher die Simulation an den realen Abläufen operiert, oder sogar mit dem Fertigungsprozess gekoppelt sein soll, desto weiter rückt deshalb die Problematik der Aktualität der Daten in den Mittelpunkt [63].

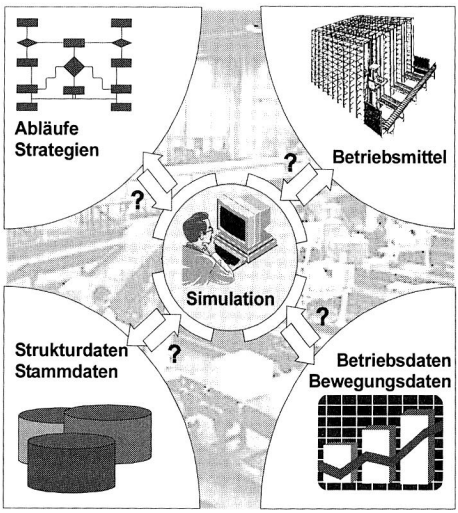
Dies gilt auch bei einer lebenszeitbegleitenden Simulationsunterstützung. Hier zeigt sich bei näherer Betrachtung, dass dabei bisher vor allem Untersuchungen und Optimierungen technisch-logistischer Parameter wie Anlagenauslastung, Durchlaufherstelldauer und Pufferbelegung im Vordergrund stehen, wobei das Optimum bezüglich zeit- und mengenmäßiger Zielvorgaben angestrebt wird. Ein so ausgelegtes Produktionssystem ist aber nicht zwangsläufig auch die wirtschaftlich beste Lösung. Um solche Untersuchungen ebenfalls durchführen zu können, erscheint es daher sinnvoll, aktuelle Daten und Ablaufsimulation zusammenzufassen und damit eine -integrierte- Kostenbewertung des Planungsobjekts durchzuführen [64,65].

Als eine wichtige Grundlage für integrierte technisch-logistische und wirtschaftliche Anwendungen der Simulation wird es vielfach angesehen, die Datenakquisition direkt am Gesamtprozess zu orientieren und damit die Qualität der Eingangsdaten für die Simulation zu erhöhen [66].

Folglich ist eine unterstützende Kommunikationsarchitektur für Simulationszwecke zu schaffen [67]. Diese muss idealerweise in der Lage sein, den Zugriff auf die verteilten Datenbestände eines Unternehmens unabhängig von der jeweiligen Betriebssystemplattform und dem Ort und der Art der Speicherung zu ermöglichen. Dies entspricht

auch der Forderung vieler (potentieller) Simulationsanwender, die Datenakquisition durch geeignete Methodiken zu vereinfachen [21, 68].

Nicht zuletzt aus dieser Forderung ist zu erkennen, dass die Simulation aufgrund der bisher mangelnden Integrationsfähigkeit in die EDV-Systeme der Unternehmen oft als alleinstehendes Werkzeug betrachtet wird und Simulationsmodelle entsprechend realisiert werden [69].



Ausgangslage

- Zur Simulation notwendige Basisdaten sind nicht in einheitlicher Form verfügbar
- Datenakquisition stellt zeitaufwendige Phase einer Simulationsstudie dar
- Fehlende Schnittstellen erzeugen häufig Insellösungen

Bedarf

- Datenakquistion an vorhandenen verteilten Prozessen orientieren
- Online-Kopplung zur schnellerern Datenbeschaffung
- Einsatz der Simulation nahe am realen Betriebsgeschehen

Abb. 8 Bedarf der verbesserten Anbindung der Simulation an das reale Planungs- und Betriebsgeschehen

Häufig wird in diesem Zusammenhang konkret die mangelnde Einbindung oder Integration der Simulation in andere Werkzeuge des CAE-Bereichs genannt (Abb. 8). [70] konkretisiert dies und nennt die fehlende Standardisierung von Datenmodellen zum Austausch simulationsrelevanter Informationen über Systemgrenzen hinweg. Wichtige Potentiale gehen somit verloren.

Aus Sicht des Simulationsanwenders hat dies zur Folge, dass ein hoher Zeitaufwand damit verbunden ist, die zur Systembeschreibung und Systembewertung notwendigen Informationen aus den Datenbeständen des Unternehmens zu 'extrahieren' [71, 72]. Die bereits genannte Umfrage unter Simulationsanwendern belegt dieses Anwendungshemmnis. Eine Reduzierung dieses Aufwands ist daher eine häufig genannte Herausforderung der aktuellen Forschung [21].

2.6 Zusammenfassung

Bereits mit den heute verfügbaren computergestützten Simulationssystemen lassen sich Produktionssysteme, Materialflüsse und organisatorische Abläufe in vielfältiger Weise modellieren. Durch Untersuchungen und Vergleich von Alternativen im Modell können Investitionen absichert, Schwachstellen in den betrieblichen Abläufen erkannt und insgesamt eine Optimierung der Produktion nachhaltig unterstützt werden.

Problematisch ist allerdings weiterhin der Einsatz der Simulation in komplexen Planungsprojekten und bei Anwendungen mit hohem Datenbedarf sowie der zeit- und betriebsnahe Einsatz eines Simulationsmodells. Was in anderen informationsbestimmten Bereichen - wie etwa dem eCommerce - ein sich klar abzeichnender Trend ist, wird bei der offensichtlich notwendigen und viel diskutierten Integration der Simulation in die betrieblichen Abläufe erst in einzelnen Bereichen vollzogen. Folglich sind derzeit erhebliche Anwendungshemmnisse zu verzeichnen: In der Praxis stellen Simulationsmodelle somit häufig noch 'Insellösungen' im unternehmensweiten Planungsgeschehen dar, was sich sehr klar auf die mangelnde Verfügbarkeit simulationsrelevanter Daten zurückführen lässt.

3 Simulationsrelevante Daten

Die Analyse der Prozesskette bei der Durchführung von Simulationsstudien identifizierte die Datenerfassung als äußerst zeit- und kostenintensiv. Zusätzlich zeigt sich, dass dieser Teilprozess durch seine Komplexität fehleranfällig ist und dies erhebliche Auswirkungen auf das Simulationsergebnis und die darauf aufbauenden Entscheidungen hat. Dies wurde als signifikantes Anwendungshemmnis der Simulation im Tagesgeschäft erkannt. Dementsprechend ist eine umfassende Integration in die betrieblichen Abläufe bisher nur punktuell zu erkennen.

Deshalb soll im folgenden eine Analyse vorgenommen werden, die den Stellenwert der Datenbeschaffung während der Durchführung von Simulationsstudien genauer untersucht. Darauf aufbauend wird anschließend der Datenbedarf entlang der einzelnen Phasen einer Simulationsstudie bewertet. Ziel ist es, im Zusammenhang mit der Ablaufsimulation die Begriffe "Daten", "Informationen" und "Parameter" zu präzisieren und einen Zusammenhang zwischen Datenquantität und -qualität herzustellen. Dies dient dann als Grundlage für die Bewertung des Standes der Technik bei der Beschaffung simulationsrelevanter Daten und des betrieblichen Informationsumfelds, in dem ein solcher Prozess stattfindet.

3.1 Einordnung der Datenakquisition innerhalb einer Simulationsstudie

Informationen sind - im Gegensatz zu allen anderen "Produktionsfaktoren" - das einzige Gut, welches i.d.R. selbst bei mehrfacher Nutzung nicht verbraucht werden kann. Sie werden deswegen auch als immaterielles Gut bezeichnet und können daher nicht direkt als Wirtschaftsgut bezeichnet werden. Vielmehr stellt erst das Nutzungsrecht an einer Information den eigentlichen Wert dar [73].

Der tatsächliche Wert richtet sich nach dem Informationsbedarf, der sich wiederum aus der Menge der notwendigen Informationen ergibt, die zur Erfüllung einer Aufgabe oder Herbeiführung einer Entscheidung notwendig sind. Somit darf der Aufwand (meist manifestiert durch angefallene Kosten) zum Erlangen einer zusätzlichen Information höchstens ihrem Wert entsprechen.

Das Problem bei der Quantifizierung des Informationswerts ist, dass sowohl Aufwand als auch Nutzen weitgehend subjektive Größen sind, die nicht in allen Entscheidungssituationen das maßgebliche Entscheidungsinstrument sein können. Somit ist die Entscheidung über Art, Menge und Güte der Informationen eine eher strategische Entscheidung (Informationsentscheidung). Diese richtet sich danach, inwieweit durch

Informationen eine anstehende Entscheidung oder Handlung verbessert werden kann. Die Konsequenz daraus ist u.a., dass Informationen als handelbare Ressourcen angesehen werden können, die mit einem gewissen Ressourcenverbrauch gewonnen, übermittelt (verteilt) und verarbeitet werden können.

Demzufolge stellt auch die Datenakquisition für ein Simulationsmodell einen vorbereitenden Arbeitsschritt dar, der die geforderte Informationsgewinnung (im Sinne von Ergebnissen) durch ein Simulationsmodell unterstützt. Dabei ist allerdings zu beachten, dass die "Datenbeschaffung" zwar vielfach als eigener Arbeitsschritt innerhalb einer Simulationsstudie definiert ist, dies jedoch praktisch nicht durchführbar ist. Vielmehr muss davon ausgegangen werden, dass ein kontinuierlicher Informationsbedarf wechselnder Intensität besteht, der sich über alle Phasen erstreckt und welcher von der Aktivität "Datenakquisition" gedeckt wird.

3.2 Daten zum Entwurf und Aufbau des Simulationsmodells

Aus den in Kap. 2.4 genannten Gründen wird im folgenden von der Vorstellung abgewichen, die Simulationsphasen als Regelkreis aufzufassen (VDI 3633 [18]). Stattdessen wird der Datenbedarf entlang der Phasen einer Simulationsstudie projekt- und informationsorientiert in die Klassen S (Struktur, Kap. 3.2.1), A (Abläufe, Kap. 3.2.2), P (Parameter, Kap. 3.2.3), D (Daten zur Experimentdurchführung, Kap.3.3.1) und E (Ergebnisdaten, Kap. 3.3.2) aufgeteilt. Ziel soll es sein, die Daten und Informationen in einer Weise und Reihenfolge zu ordnen, wie sie später in einem Simulationsmodell verwendet werden.

3.2.1 Ermittlung von Daten zur Strukturbildung: Datenklasse S

In den frühen Phasen einer Simulationsstudie - in diesem Fall nachdem Aufgaben und Ziele festgelegt sind - werden zunächst Daten über die statischen Eigenschaften des Systems gesucht. Häufig handelt es sich hier um eine intensive Zusammenarbeit zwischen Planer und Simulationsexperten, bei der das Umfeld der Studie genauer analysiert wird und die Systemgrenzen fixiert werden. Hierzu gehören Organisationsbeschreibungen, Mitarbeiter und Maschinen, die für die Durchführung von Prozessen notwendig sind. Weiterhin werden die Relationen zwischen diesen Einzelobjekten festgelegt (Abb. 9).

Grundlage sind bei diesem Punkt meist CAD Unterlagen in gedruckter oder elektronischer Form. Hierbei handelt es sich im Anwendungsfeld von Produktionssystemen um topologische Beschreibungen. Erfasst werden somit produkt-, prozess- und anlagenbezogene Informationen, die statisch einem Standort zur Leistungserbringung zugeordnet werden. Diese legen fest, wo ein Prozessschritt ausgeführt wird und auf

welchen Wegen das zu bearbeitende Gut zum Bearbeitungsschritt gelangt. Um auch den Anforderungen an die Prozessgestaltung während der Planungsphase zu genügen, ist dabei vorzusehen, dass die gewonnenen Informationen jederzeit abstrahiert, detailliert, modifiziert oder mit zeitlichen Gültigkeiten versehen werden können (Abb. 10).

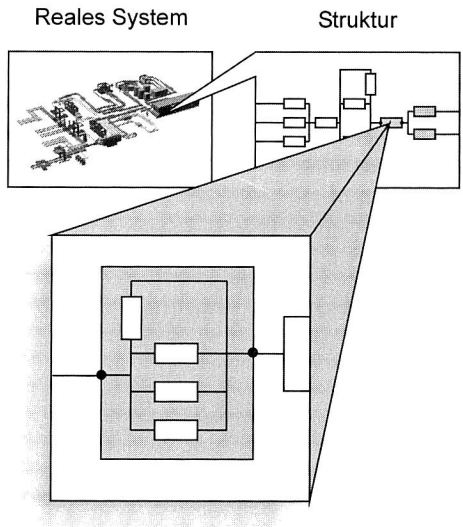


Abb. 9 *Beschaffung von Strukturinformationen, die einem Element statisch zugewiesen werden können*

Die Daten und Informationen zur Struktur sind somit die wirkungsmäßige Anordnung der Komponenten in einem System, von dem das Simulationsmodell eine vereinfachte Nachbildung darstellen soll. Die Aufgabe des Simulationsexperten ist es daher, diese Aufbaustruktur des vorliegenden technischen Systems mit hinreichender Genauigkeit abzubilden. Systemtechnisch ergibt sich hieraus später die notwendige funktionale Abgrenzung des untersuchten Bereichs von anderen Unternehmensbereichen.

Dazu werden Modellbeschreibungen herangezogen, die standardisierte oder benutzerdefinierte Symbole als Bausteine zur Darstellung der Aufbaustruktur des Modells, ergänzt durch textuelle Beschreibungen, nutzen.

Verbreitete Beschreibungsmethoden sind Notationen von Warteschlangenmodellen, das *Reference Model for Shop Floor Production* [74] oder das IDEF-0 Modell [75]. Entscheidend für die Daten zur Strukturbildung ist, dass die potentielle Abfolge be-

trieblicher Aktivitäten an Operanden sowohl graphisch als auch formal beschrieben werden kann. Festzulegen sind also Operanden zu

- Materialfluss,
- Energiefluss,
- Ressourcen (Personal, Maschinen usw.),
- Steuerungs-/Kontrollinformationen und
- Daten,

durch die die 'generischen Prozesse' Transportieren, Bearbeiten (Transformieren), Prüfen und Lagern ermöglicht werden.

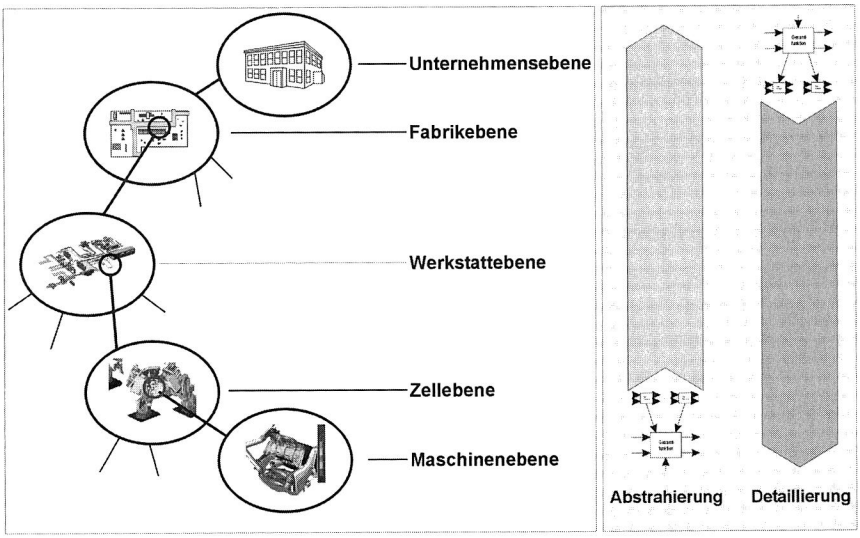


Abb. 10 Beschreibungsmöglichkeiten zur schrittweisen Abstraktion und Detaillierung bei der Informationsbeschaffung

Zusätzlich ist eine Unterscheidung zwischen Informationen und Daten notwendig, um bereits bei der Aufbaustruktur einen hierarchischen Ansatz verfolgen zu können: Kontrollinformationen sind dabei Informationen, die von einer höheren Ebene an eine tiefere Ebene gegeben werden, womit Aktivitäten auf nachgeordneten Strukturelementen veranlasst werden.

Im Gegensatz dazu werden Daten als Bestandteil eines Informationsflusses aufgefasst, der horizontal stattfindet, d.h. einen Austausch von Daten innerhalb einer Ebene beschreibt. Die Strukturierung verlangt keine Definition von Datenstrukturen, son-

dern nur die Anforderung an Schnittstellen zwischen Systemelementen, welche sich aus dem realen oder geplanten System, den Untersuchungszielen und vorgegebenen Randbedingungen ergeben. Es handelt sich also um einen Rahmen, innerhalb dessen die später zu beschreibenden zeit- und zustandsabhängigen Abläufe stattfinden (Abb. 11).

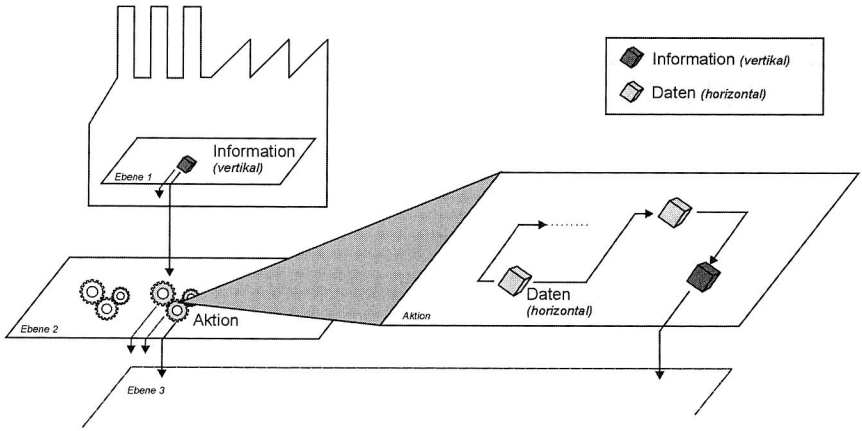


Abb. 11 Begriffsbestimmung des horizontalen und vertikalen Informationsflusses

3.2.2 Ermittlung von Daten zur Ablaufbeschreibung: Datenklasse A

Das wichtigste Merkmal eines Ablaufsimulationsmodells ist das dynamische Verhalten des Modells während der Simulationslaufzeit. Im Gegensatz zu anderen 'statischen Simulationsmethoden' finden hier Veränderungen 'auf' der Struktur (Zustandsänderungen) aufgrund des Modellzustandes und der Zeit statt. Demzufolge muss eine Ablaufbeschreibung vorhanden sein, die alle zeit- und situationsabhängigen Zustandsabfolgen festlegt. Dies ist die Grundlage der Beschreibung des Systemverhaltens.

Typischerweise erfolgt die Ablaufbeschreibung nachdem die Systemstruktur ermittelt wurde. Es wird festgelegt, wie sich ein Systemelement aufgrund seiner jeweiligen Eingangsoperanden und der Zeit verhält. Es wird also eine (in-)formelle Aussage über das interne Verhalten des jeweiligen Systemelementes gemacht. Hierzu gehören

- Transformationsfunktionen von Ein- und Ausgangsgrößen
- logische Bedingungen
- Entscheidungsregeln
- Aktionen und Aktionssequenzen als funktionale Organisationselemente von komplexen Prozessen.

Das hierbei anzutreffende Verhalten ist sowohl von der Charakteristik als auch von den dazu notwenigen Ablaufbeschreibungen sehr unterschiedlich. Mit Einschränkung auf die ereignisdiskrete Simulation (Ablaufsimulation) können als grobe Klassifikation ein diskretes, stochastisches und/oder deterministisches, zeitabhängiges, ereignis- oder aktivitätsorientiertes Verhalten vorgefunden werden. Demgemäss variieren die Methoden zur Ablaufbeschreibung und damit konsequenterweise auch die zur Beschreibung notwendigen Daten. Die Ablaufbeschreibung muss weiterhin der Strukturdetaillierung folgen können.

Darüber hinaus unterliegt eine Ablaufbeschreibung bei der Entwicklung eines Simulationsmodells dem Prozess der funktionalen Dekomposition [76], was im folgenden als Hierarchisierung bezeichnet wird. Hiermit folgt die Ablaufbeschreibung den Veränderungen der Struktur.

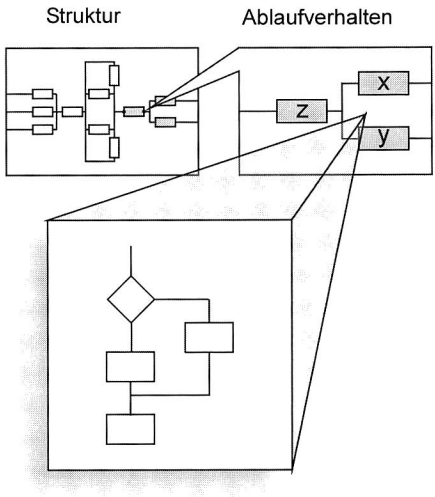


Abb. 12 Definition des Ablaufverhaltens auf der Grundlage der ermittelten Struktur des betrachteten Systems

Die Methoden zur Ablaufbeschreibung zielen entweder auf die Nutzung einer Programmiersprache oder vorgefertigter Funktionsblöcke (Bausteine) eines Simulators ab. Allgemein kann aber davon ausgegangen werden, dass folgende Beschreibungsmethoden (Abb. 13) anerkannt sind:

- Zustandsübergangsdiagramme
- Blockdiagramme
- Programmablaufpläne

- Entscheidungstabellen (Regeln, Strategien)
- Petri-Netze
- Warteschlangenmodelle
- Automaten
- Programme, Funktionen, Algorithmen

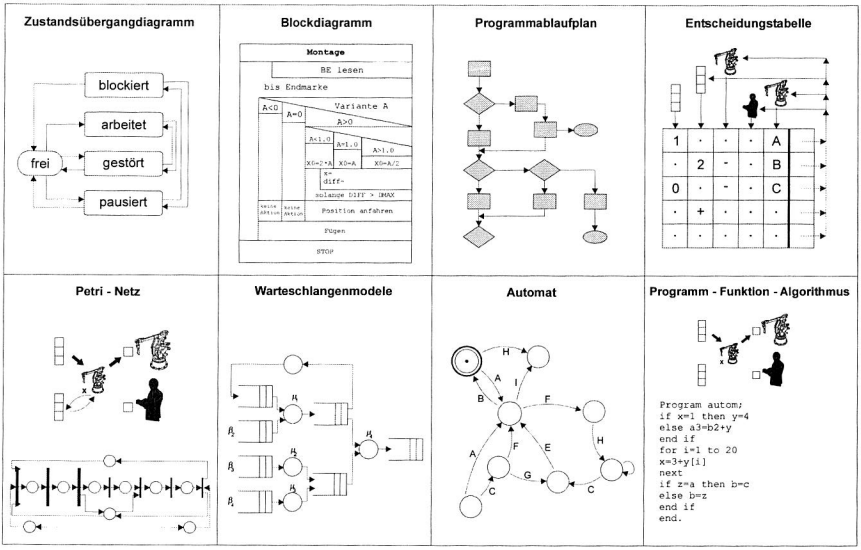


Abb. 13 Typische Methoden der Ablaufbeschreibung eines konzeptionellen Modells für Ablaufsimulationen

Diesen Beschreibungen wird dabei zunächst der Stellenwert einer Notation eingeräumt werden, mit deren Hilfe Planer und Simulationsexperte kooperativ eine sukzessiv detailliertere Ablaufbeschreibung erarbeiten werden. Daraus entsteht das konzeptionelle Modell. Startpunkt dieser Beschreibung ist die o.g. Systemstruktur, die durch die Ablaufbeschreibung und die vorherrschenden Wirkzusammenhänge erweitert wird (Was passiert wann unter welcher Bedingung?). Beispiel: In einem Materialflusssystem (Struktur) muss an einer Verzweigung der weitere Fahrweg eines Fahrzeugs bestimmt werden. Die Datenklasse S gibt zunächst Auskunft über den Aufbau des Weges und der vorhandenen Verzweigung. Diese Wegewahl an der Verzweigung könnte nun einerseits von der Beladung des Fahrzeugs selbst, aber auch von der Belegung potentieller Wegstrecken oder dem vorgegebenen Fahrkurs abhängen.

Abhängig vom geforderten Wirkzusammenhang können hieraus nun die notwendigen Informationen identifiziert werden, die auf der gewählten Abstraktionsebene für die anstehenden Entscheidungen verknüpft werden müssen. Die ermittelten Ablaufbeschreibungen müssen anschließend vom Simulationsexperten im verwendeten Simulationssystem implementiert werden. Unter diesem Arbeitsschritt kann gleichzeitig eine Vereinheitlichung aller gesammelten Ablaufinformationen verstanden werden.

Aufgrund der Komplexität von Simulationsmodellen werden hier auch hohe Anforderungen an die Präzision der Implementierung gestellt. Daraus ergibt sich der erweiterte Nutzen, dass der Simulationsexperte detaillierte Fragen zum System stellen muss, die von den Planern aufgrund fehlender Primärdaten nicht direkt beantwortet werden können und somit zu einem weiteren "*Durchdenken der Aufgabe und zu genauerer Erfassung der notwendigen Daten zwingt*" [77].

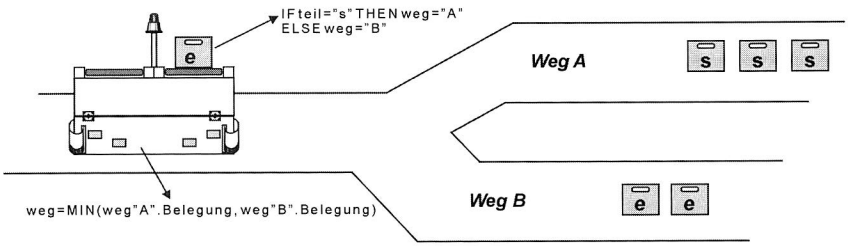


Abb. 14 Ablaufbeschreibung eines FTS-Systems als Beispiel für Informationen in der Datenklasse A

Besonders während früher Planungsphasen können so kritische Punkte in der Administration, Disposition und Steuerung des betrachteten Systems rechtzeitig erkannt werden [78].

3.2.3 Ermittlung von Daten zur Erfassung von Parametern: Datenklasse P

Einer der Hauptgründe des Einsatzes von Simulationen ist die Komplexität der realen Vorgänge mit ihren stochastischen, heuristischen, deterministischen und nicht linearen Prozessen.

Die wichtigsten Quellen dieser Komplexität sind die Komponenten des Systems, die die "Treiber" der Systemdynamik darstellen. Beispiele für solche Quellen sind Aufträge, Ankunftsraten, Arbeitszeitmodelle, Taktzeiten, Rüstzeiten und Störhäufigkeiten (Abb. 15). Daher ist zwischen der Ablaufbeschreibung und dem zu betrachtenden System eine weitere Beziehung herzustellen, die Zufallsmodelle und Beobachtungen des realen Systems parametrisch abbildet.

Als Gegensatz zu gängigen stochastischen Anwendungen geht es im Zusammenhang mit der Simulation jedoch nicht darum, ein vollständiges und perfektes Parametermodell (Eingangsdatenmodell) zu erstellen; vielmehr erfolgt die Parametererfassung ebenfalls unter den bereits genannten Gesichtpunkten der Abstraktion und Vereinfachung [79]. Die hinreichende Gültigkeit kann jedoch durch erprobte Methoden (z.B. Sensitivitätsanalysen oder Anpassungstests) sichergestellt werden.

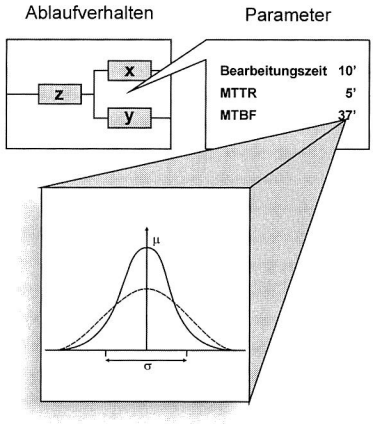


Abb. 15: Beschaffung von Informationen zur Datenklasse P auf der Basis der bereits akquirierten Ablaufbeschreibung

Der Datenbedarf entsteht somit in zwei unterscheidbaren Bereichen des zu untersuchenden Systems. Einerseits sind die Operanden an der gezogenen Systemgrenze zu berücksichtigen, etwa Kundenaufträge. Andererseits ist für jedes Element der Ablaufbeschreibung der Datenbedarf zu decken, da erst die Ablaufschreibung gemeinsam mit den entsprechenden Parametern eine vollständige Charakterisierung eines Systemelements zulässt. Beispielweise kann das Verhalten zweier Bestücker vom Ablauf her zunächst gleich sein, jedoch durch unterschiedliche Parameter (etwa der Bestückumfang oder die Wahrscheinlichkeit von Nebenzeit durch nachlässiges Splicen des Maschinenpersonals) eine vollkommen andere Ausprägung ergeben (Abb. 16).

Die hierzu notwendigen Informationen sind in geeigneter Weise aus dem realen Betriebsgeschehen oder dem Planungsprozess zu ermitteln. Dies sind einerseits zeitraumbehaftete Informationen, die die Zustandsdauer (Arbeitend, Blockiert, Pausiert, Gestört usw.) von Maschinen, Ressourcen und Personal beschreiben und andererseits zeitpunktbehaftete Informationen, welche die Transitionen des Systemzustands, etwa Störungen, erfassen.

Die eigentlichen Daten und zu erzeugenden Parameter sind so vielfältig wie die Beschreibungsmöglichkeiten der Abläufe. Der Datenumfang und die notwendige Datenqualität muss mindestens der Ablaufbeschreibungsqualität entsprechen. In diesem Sinne folgt also die Komplexität des zu erhebenden Datenmaterials zwangsläufig der berücksichtigten Komplexität der Ablaufbeschreibung. Dies bedeutet einerseits, dass nicht verfügbare Daten detaillierte Ablaufbeschreibungen in eine höhere Abstraktionsebene zwingen und andererseits detailliertes Datenmaterial ohne Nutzungsmöglichkeit in der Simulationsstudie letztlich unwirtschaftlichen Ballast darstellt.

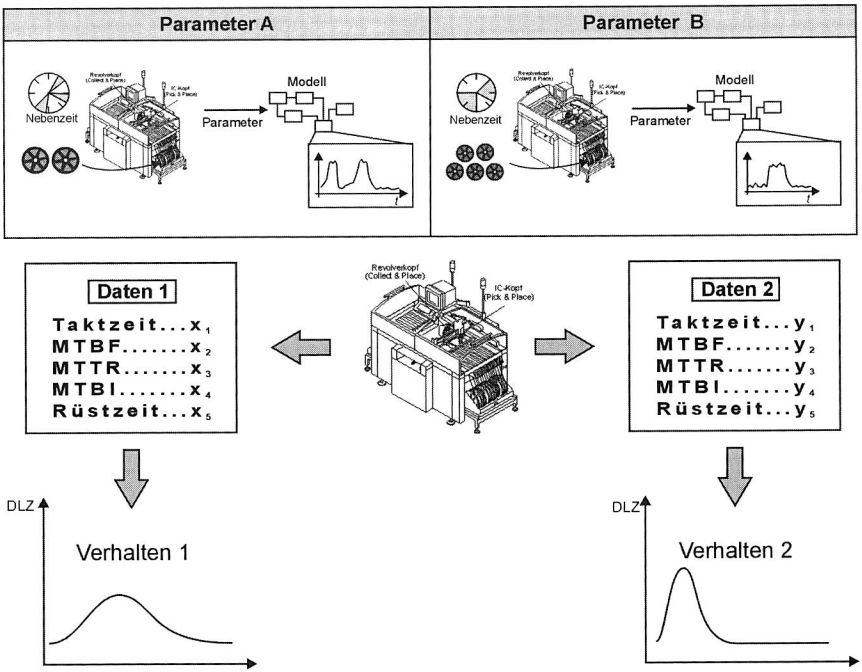


Abb. 16: Unterschiedliche Parameter bestimmen das charakteristische Verhalten eines Systemelements bei sonst gleicher Beschreibung

Aktuelle Bestrebungen richten sich daher auf eine zielgerichtete Datenbeschaffung. Die Diskussionen über Referenzmodelle beim Simulationseinsatz zeigen, dass hier für unterschiedliche Einsatzgebiete der zu erwartende Datenbedarf branchen- oder fachspezifisch mit hoher Praxisrelevanz definiert werden kann [80], wie es beispielsweise bei der Modellierung von Geschäftprozessen im Rahmen der SAP Referenzmodelle bekannt ist [81]. [82] zeigt hierzu den aktuellen Stand der Technik.

Ein weiterer Ansatz zur strukturierten Beschaffung von Systemdaten innerhalb eines Anwendungsfelds, der für den Bereich der hochautomatisierten Systeme der (Elekt-

ronik-) Produktion zunehmende Bedeutung gewinnt, wird im *Semiconductor Equipment and Materials International* Standard SEMI E10-0699 beschrieben ("*Standard for Definition and Measurement of Equipment Reliability, Availability and Maintainability (RAM)*") [83]. In diesem Standard werden inhaltliche und begriffliche Definitionen vorgestellt, die eine umfassende Erfassung und Bewertung von Systemzuständen dieses Produktionstyps erlauben. Diese sind ohne weitere Aufbereitung dazu geeignet, als Simulationsparameter herangezogen zu werden.

Es werden sechs Betriebszustände aufgeführt, in die die beobachteten Perioden eines zu beobachtenden Elements fallen müssen.

- Produktiv: Das Element erfüllt seine vorgesehene - produktive - Aufgabe.
- Standby/Wartet: Das Element ist betriebsbereit, wird jedoch nicht betrieben.
- Experiment/Test: Das Element könnte produktiv sein, wird jedoch im Rahmen von Prozess-, Geräte- oder Steuerungsoptimierung ("engineering") eingesetzt.
- Geplante Nebenzeit/Wartung: Das Element ist nicht produktiv, beispielsweise aufgrund vorgesehener Wartungen, Tests, Rüstvorgänge.
- Ungeplante Nebenzeit/Störung: Das Element befindet sich in einem Zustand, der unregelmäßig auftritt und es in einen nicht produktiven Zustand versetzt. Hierzu gehören Störungen oder Instandhaltungen, etwa das Ergänzen oder Ersetzen von Hilfsstoffen.
- Ungeplanter Zustand/Unterbrochen: Das Element ist ohne technischen Grund inaktiv, etwa durch arbeitsfreie Schicht, Urlaub oder Wochenenden.

Den Zusammenhang zwischen den Zeitanteilen verdeutlicht Abb. 17.

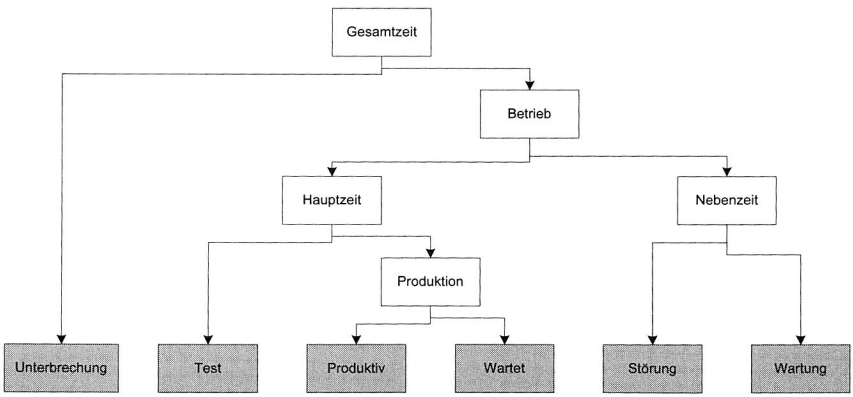


Abb. 17: Zustandsbezogene Informationen zur Datenklasse P gemäß SEMI-E10

Neben dieser zustandsbezogenen Strukturierung der Zeitanteile ist weiterhin wichtig, die zu ermittelnden Parameter durch eine geeignet strukturierte Datensammlung zu erhalten. RAM klassifiziert nach Zuverlässigkeit, Verfügbarkeit, Wartung und Auslastung mit 16 einzelnen Kenngrößen. Dies zeigt Tabelle 1.

Zuverlässigkeit	Verfügbarkeit	Wartung	Auslastung
MTBI: mean time between interrupts	equipment-dependent uptime [%]	MTTR(f): mean time to repair - failure	operational utilization [%]
MTBF: mean time between failures	supplier-dependent uptime [%]	MTTR(i): mean time to repair - (interrupt)	total utilization [%]
MTBA: mean time between assists	operational time [%]	MTOL: mean time off-line	
MCBI: mean cycles between interrupts		equipment dependent scheduled downtime [%]	
MCBF: mean cycles between failures		supplier dependent scheduled downtime [%]	
MCBA: mean cycle between assists			

Tabelle 1 Strukturierung von Zeitanteilen nach [83, Tabelle1]

Grundvoraussetzung ist jedoch in jedem Fall, dass Grunddaten, Informationen und letztlich Parameter in ausreichender Menge, Qualität und Signifikanz vorhanden und erreichbar sind, d.h. tatsächlich erfasst werden (können/dürfen). Dies ist nicht in jedem Fall möglich: Beispielsweise existiert das zu untersuchende System noch nicht oder eine Datenbeschaffung nicht erlaubt ist, etwa weil die Erfassung von personenbezogenen (Leistungs-) Daten eingeschränkt ist. In Deutschland stellen hier das Bundesdatenschutzgesetz (BDSG, hier z.B. §4 Abs. 2) [84], Tarifverträge und Betriebsvereinbarungen erhebliche Restriktionen dar.

In anderen Fällen reicht die Zeit (oder eine andere Ressource) für eine umfangreiche Datenbeschaffung nicht aus. Dies betrifft nicht nur die Datenverfügbarkeit als notwendige Bedingung, sondern ist auch von großem Interesse, um das Datenmaterial anhand statistischer Methoden hinsichtlich seiner Zuverlässigkeit (z.B. Konfidenzintervalle) bewerten zu können, wie etwa in [85] beschrieben wird: Beispielsweise kann bei einer Betriebszeit einer Maschine von 1200 Stunden und sechs Fehlerereignissen davon ausgegangen werden, dass die mittlere Zeit zwischen zwei Ausfällen (MTBF) 200 Stunden beträgt (1200 / 6 [h]), dass 90% Konfidenzintervall liegt jedoch zwischen 114 und 380 Stunden ($ku_{6;0.9} = 0.57$, bzw. $ko_{6;0.9} = 1.904$). Eine Verbesserung lässt sich in solchen Fällen nur erreichen, wenn die Stichprobe entsprechend größer ausfällt, d.h. mehr Daten zur Verfügung stehen (Abb. 18). Die Statistik nennt hier die Faustregel, dass sich die Datenqualität nur mit quadratisch steigendem Auf-

wand (Datenquantität) verbessern lässt. Ein Ziel dieser Arbeit wird es sein, dies ohne signifikant höheren Aufwand für den Simulationsanwender zu erreichen.

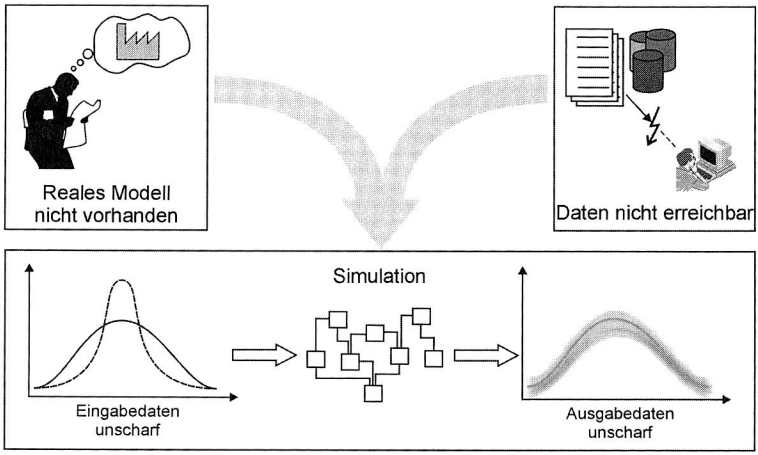


Abb. 18: Typische Ursachen für mangelhafte oder unzureichende Daten sowie die Konsequenz für die Ergebnisqualität bei Simulationsstudien

Über die Daten zur Bewertung der Verfügbarkeit und Zuverlässigkeit hinaus sind in der Datenklasse P die Parameter bezüglich Taktzeiten, sowie zur Systemlast zu erfassen. Dabei können die Daten zur Taktzeit als weitere Aufschlüsselung des Zeitanteils "produktiv" aufgefasst werden.

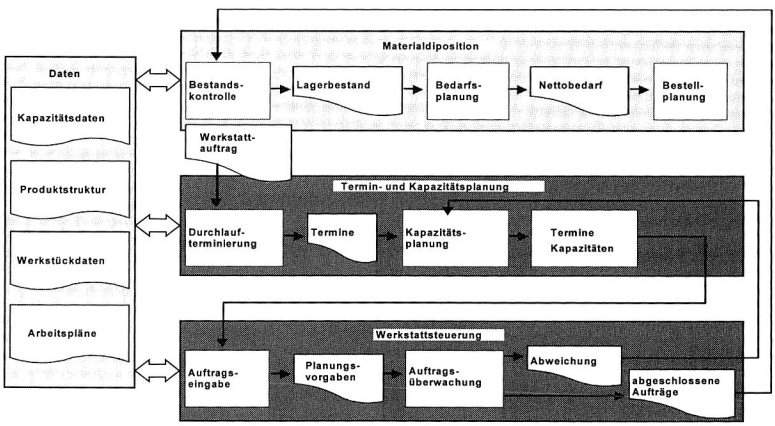


Abb. 19: Potentielle Informationen aus der Auftragsabwicklung in der Produktionsplanung und -steuerung (nach [86])

Die Systemlastdaten können typischerweise mit unterschiedlich genauen Lastprofilen, z.B. in tabellarischer Form, erfasst und bereitgestellt werden. Dieser Schritt ist meist schnell und einfach abzuhandeln, da die Systemlast als Primardatum aus der Produktionsplanung und -steuerung, der Betriebsdatenerfassung und/oder Fertigungsleitständen bei hochautomatisierten Produktionssystemen häufig direkt bereitsteht. Abb. 19 verdeutlicht die potentiellen Datenquellen, welche innerhalb einer PPS-Funktionalität üblicherweise verwendet werden.

3.3 Daten zur Vorbereitung und Durchführung der Simulationsexperimente

Für den Auftraggeber einer Simulationsstudie ist die Durchführung von Simulationsexperimenten von zentralem Interesse, da meist erst in diesem Schritt die Investitionen in das Simulationsmodell während des Planungs- und Entscheidungsprozesses gewinnbringend eingesetzt werden können. Daher muss untersucht werden, inwieweit Daten zur Experimentplanung benötigt werden und welche Daten bei den Simulationsexperimenten anfallen.

3.3.1 Ermittlung von Daten zur Experimentdurchführung: Datenklasse D

Die Abbildung eines komplexen Systems hat auch modellseitig eine entsprechende Komplexität zur Folge. Somit ist für die Analyse des Systemverhaltens im Rahmen einer Simulationsstudie eine zielgerichtete, d.h. geplante, Vorgehensweise unerlässlich. Weil zudem die zur Verfügung stehende Zeit eine beschränkte Ressource ist, muss die Menge, Aussagekraft und Qualität der Ergebnisdaten hinsichtlich ihrer statistischen Zuverlässigkeit durch geeignete Vorgehensweisen sichergestellt werden. Die Ergebnisdaten hängen in jedem Fall von den Untersuchungszielen, den Eingangsdaten und den zu beachtenden Randbedingungen ab. [87] dokumentiert entsprechende Überlegungen und Methoden.

Im Zusammenhang mit der vorliegenden Arbeit sind dabei die eigentlichen Experimentdurchführungsmethoden (wie z.B. in [87] beschrieben) von untergeordneter Bedeutung, da diese in Simulationsstudien üblicherweise durch Werkzeuge realisiert werden, die nicht direkt der Methode 'Simulation', sondern eher der Optimierung zugerechnet werden sollten.

Dies hat zur Folge, dass die Experimentplanung innerhalb eines Regelkreises ('finden eines Optimums innerhalb der potentiell untersuchbaren Möglichkeiten') zwischen den formulierten Zielen sowie den Ein- und Ausgangsgrößen des Simulationsmodells agiert. Die Datenklasse D umfasst daher alle Eingangsgrößen, die für die Initialisierung eines Simulationsmodells und zur Durchführung eines Simulationsex-

periments gebraucht werden und zusammen mit der Menge der Ergebnisdaten (Datenklasse E, Kap. 3.3.2) ein Experiment beschreiben.

3.3.2 Ermittlung von Daten aus Simulationsergebnissen: Datenklasse E

Die Bewertung von Daten aus Simulationsexperimenten geht zunächst davon aus, dass Simulationsergebnisse eine Grundlage zur Systemanalyse liefern. Darüber hinaus ist von Bedeutung, dass im betrieblichem Umfeld die Simulation zunehmend als strategisches Werkzeug davon geprägt ist, langfristig, regelmäßig und durchgängig eingesetzt zu werden - Simulationsergebnisse sollen zukünftig nicht mehr isoliert verwendet werden.

Dementsprechend kann die Bereitstellung von Ergebnissen jederzeit als Zwischenschritt (Zwischenergebnis) angesehen werden, da diese Daten als Eingangsgrößen für spätere Simulationsstudien dienen können (Fokus: Langfristigkeit). Unter diesem Blickwinkel können Ergebnisdaten einer Simulationsstudie als Eingangsdaten einer Folgestudie aufgefasst werden (Fokus: Durchgängigkeit), die bei geeigneter Verwaltung direkt als Primärdaten zur Verfügung stehen (Fokus: Regelmäßigkeit).

Daraus ergibt sich letztlich eine erweiterte Anforderung an die Datenkopplung: Daten, die aus Simulationsergebnissen ermittelt worden sind, müssen gespeichert und zusammen mit Metainformationen verwaltet werden: Somit ist neben der Wiederverwendung der Eingangsdaten (also die Datenklassen S, A und P) auch die Wiederverwendung der Simulationsergebnisse (hier die Datenklasse E) in den Anforderungskatalog der offenen Datenkopplung aufzunehmen.

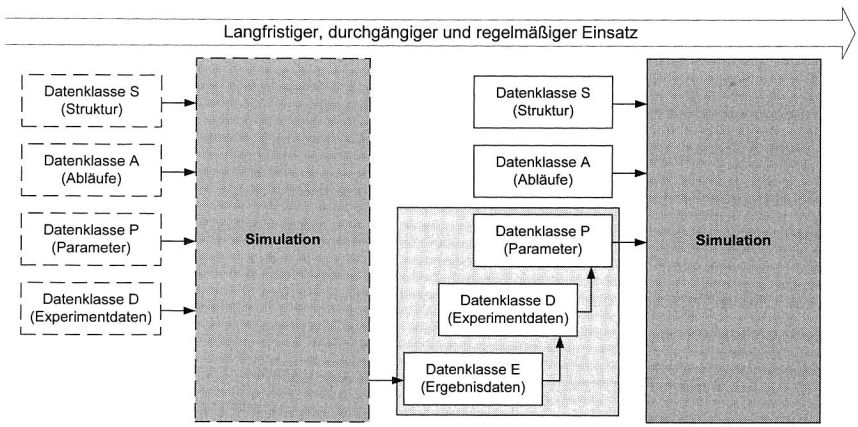


Abb. 20: Zusammenfassung der Datenklassen P, D und E bei einem in die betrieblichen Abläufe integrierten Simulationseinsatz

Technisch können die Ergebnisdaten damit als Eingangsdaten einer Simulationsstudie aufgefasst werden, die um Informationen hinsichtlich des zugrundeliegenden Simulationsmodells ergänzt werden müssen. Weil diese Ursprungsinformation ("Aus welchem Simulationsmodell kommen die Ergebnisse?") als gleichwertig zur Standortidentifikation der Informationen aus Datenklasse P ("Aus welcher Datenquelle kommen die Parameter?") aufgefasst werden soll, sind die Datenklassen P und E unter dem Aspekt der Datenakquisition als äquivalent anzunehmen (Abb. 20): Die Ergebnisse eines Simulationsexperiments können als Eingangsgröße(n) für ein weiteres Simulationsexperiment oder -modell dienen.

Dies entspricht zunehmend auch der Praxis, bei der beispielsweise von einem Zulieferer eine Simulation gefordert wird, die Aussagen über Lieferfähigkeit bereitstellt. Diese Aussage wäre der o.g. Argumentation folgend primär ein Ergebnis, welches anschließend beim Kunden als Eingangsparameter beispielsweise für eine simulationsunterstützte Gestaltung der Eingangslagerkapazitäten herangezogen wird.

3.4 Zuordnung der Akquisitionsklassen zu den Phasen nach VDI 3633-1

Die klassische Vorgehensweise bei der Beschaffung der o.g. Daten sieht eine weitgehend sequentielle Vorgehensweise vor, wobei durch ein konzeptionelles Modell alle Eingangsdaten nacheinander ermittelt werden. Dies wird als Stand der Technik in der VDI Richtlinie 3633 festgehalten. Durch die steigende Leistungsfähigkeit und die weiterentwickelten Methoden ist zunehmend der Trend zu verzeichnen, dass die genannten Klassen nicht getrennt voneinander mit Informationen versorgt werden. Vielmehr sind moderne Simulationssysteme in der Lage, die genannten Datenklassen in einem Arbeitsschritt zu erfassen und als Beschreibung eines Simulationsmodells zu verwenden. Die erwähnten Referenzmodelle belegen in diesem Zusammenhang punktuell die Leistungsfähigkeit der zugrundeliegenden Systeme. Hierdurch besteht die Möglichkeit, dass die Grenze zwischen dem (konzeptionellen) Modell des betrachteten Systems und dem implementierten Simulationsmodell aufgehoben wird.

Voraussetzung für die erfolgreiche Nutzung solcher Modellierungsmethoden können entweder bausteinorientierte Simulationssysteme, Referenzmodelle oder jede andere Art von Modellierungshilfsmitteln, z.B. UML [88] oder IDEF0 [89] sein. Eine standardisierte Beschreibung, wie sie in anderen Bereichen der Fertigungstechnik punktuell bereits eingesetzt wird, beispielsweise IEC1131 [90] oder STEP [91], hat sich im Bereich der Ablaufsimulationstechnik jedoch noch nicht etablieren können. Die vorher vorgestellte Klassifikation soll einen Schritt in diese Richtung darstellen. Sie soll einen systematischen Ansatz bieten, um den Datenbedarf für eine Ablaufsimulations-

studie besser darstellen zu können. Auf die Darstellung von Datendetails (Atomen) wird verzichtet, um den konzeptionellen Charakter zu betonen. Die Ergebnisse von Kap. 5 werden zeigen, dass dies nicht als Einschränkung zu sehen ist, da sich simulationsrelevante Daten meist über die Metainformation "Ursprung" identifizieren und nutzen lassen.

Aus Sicht der Datenbeschaffung ist der wichtigste Unterschied zwischen der vorgehend beschriebenen Klassifikation und dem Vorgehen nach VDI 3633, dass die dort genannten Aktivitäten (Problemdefinition, Zieldefinition, Datenbeschaffung, Modellbildung usw.) als weitgehend abgeschlossene Arbeitspakete verstanden werden. Iterationsschleifen werden dabei als Ausdruck von fehlenden, fehlerhaften oder qualitativ unzureichenden Informationen betrachtet. In der Praxis zeigt sich jedoch, dass - im Gegensatz zur Auffassung des klassischen Projektmanagements und des Softwareengineerings - bei Simulationsprojekten das Problem- und Systemverständnis meist noch nicht so ausgeprägt ist, wie es für ein adäquates Lastenheft notwendig wäre. Da die Simulation als Werkzeug im Endergebnis nicht nur einen Beitrag zur Problemlösung leistet, sondern bereits bei der Gewinnung von Wissen ("Know-how") über das System projektbegleitend eingesetzt werden soll, sind solche Iterationsschleifen unvermeidlich und sogar gewünscht - die Lern- oder Erkenntniskurve kann im Gegensatz zum Projektstand als eine monoton steigende Funktion aufgefasst werden (Abb. 21).

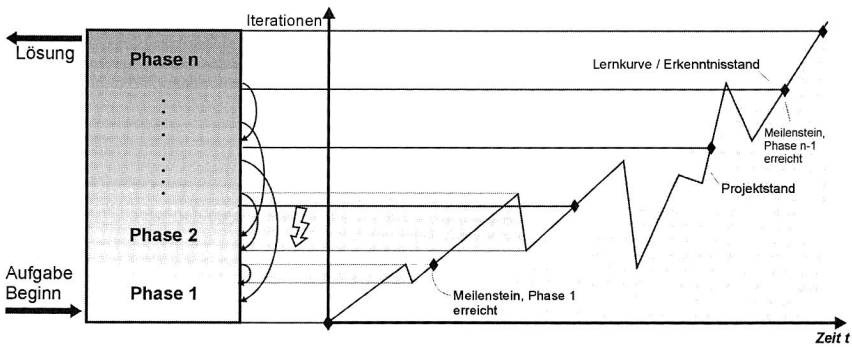


Abb. 21: Projektfortschritt und Erkenntnisstand während der Durchführung einer Simulationsstudie

Obwohl die vorgestellte Klassifikation somit eine Sicht anbietet, die sich an dem Methodenverständnis der Anwender orientiert, muss nun geprüft werden, ob diese Klassifikation zur VDI 3633 Richtlinie - als allgemein akzeptierter Stand der Technik nach VDI 1000 - hinsichtlich der Projektorganisation und -durchführung widerspruchsfrei ist. Dies soll durch eine Transformationsmatrix erfolgen, in der die 'klassischen' Pha-

sen sowie die Datenklassen miteinander unter dem Gesichtpunkt des Datenbedarfs verknüpft werden.

Zuordnung Phase → Datenklasse	**Datenklasse S (Struktur)**	**Datenklasse A (Abläufe)**	**Datenklasse P (Parameter)** ≡ Datenklasse D/E
1. Problembeschreibung	(-)[5]	(-) [5]	- [5]
2. Zieldefinition	-	-	(+)
3. Modellkonzept	**+**	**+**	**+**
4. Datenbeschaffung	+ [4]	+ [4]	**+**
5. Modellbildung	**+**	**+**	+
6. Validation	(+) [3][5]	(+) [3][5]	**+**
7. Experimentplanung	(-) [1]	(-) [1]	**+**
8. Experimente	(+) [1]	(+) [1]	**+**
9. Präsentation	(-)	(-)	**+ [2]**

Legende: *+ geeignet (+) bedingt geeignet (-) eher ungeeignet - ungeeignet*
*(Der technische Schwerpunkt der einzelnen Phase wird **fett** geschrieben)*

[1] wird von externen Werkzeugen festgelegt / gesteuert
[2] aufgrund der angenommenen Klassenäquivalenz sinnvoll
[3] Einschränkung, da Verifikation, Validation und Testen (VV&T) hohen manuellen Aufwand erfordern, tw. Heuristiken
[4] Einschränkung aufgrund a priori nicht exakt definierbaren Beschreibungsmöglichkeiten
[5] Aufgrund des teilweise informellen Charakters

Tabelle 2: Zuordnung der erarbeiteten Datenklassen zu den durchzuführenden Phasen einer Simulationsstudie gemäß VDI 3633

Wie aus Tabelle 2 zu entnehmen, können die Aufgaben innerhalb einer Simulationsstudie durch die Datenklassen in ihrer vorgelegten Form befriedigt werden. Die detaillierte Aufschlüsselung der notwendigen Informationen zu den Datenklassen wird ein wichtiges Merkmal des zu entwerfenden Architekturkonzepts sein.

Es ergeben sich bereits einige Anforderungen und Schwerpunkte, die beim Entwurf zu berücksichtigen sind: Besonders ist zu erkennen, dass die Datenklassen für Struktur und Abläufe häufig mittelbar den Phasen zugeordnet werden können. Dies ist in der vorgehend genannten Bandbreite der verwendeten Informationen begründet. Im Gegensatz dazu kann die Datenklasse für Parameter unmittelbar den einzelnen Phasen zugeordnet werden. Dabei ist festzuhalten, dass die Datenklasse P direkt aus den Datenklassen S und A bestimmt wird. S und A werden daher konzeptionell als Metainformation zur Strukturierung der Informationsbestände von Datenklasse P herangezogen.

Die Datenkopplung orientiert sich damit tatsächlich an den notwendigen Daten (Datenklasse P/D/E). Sie wird in einer Weise genutzt werden, wie in [92] vorgeschlagen wurde: Die Autoren gehen von einer IDEF-0 formalisierten Strukturbeschreibung aus. Aus dem so entstehendenden "entity model" wird dann ein 'mapping table' erstellt, der letztlich die notwendigen Daten systematisch aus Datenbanken (also explizit keinen realen Prozessdatenquellen) zu erfassen hilft. Strukturen und Abläufe werden dabei als Hilfsmittel zur systematisch durchgeführten Datenbeschaffung angesehen.

Aus Tabelle 2 ist weiterhin zu entnehmen, dass auch die Datenklassen unterschiedlichen Phasen unterworden sind. Der Grund dafür ist die methodische Trennung zwischen Investition (Modellbildung) und produktivem Einsatz des Simulationsmodells. Während in den Phasen eins bis fünf noch kein Simulationsmodell verfügbar ist, überwiegen die eingebrachten Ressourcen meist den Ertrag. Erst in Phase sieben und den folgenden Arbeitsschritten werden die Ergebnisse des betriebenen Aufwands für den Auftraggeber sichtbar und verwertbar.

Diese Trennung verdeutlicht zwei unterschiedliche Anwendungsfelder, die in der Definition der Anforderung als 'Modellierung' und 'Betrieb' bezeichnet werden und in den Kapiteln 4.1.1 und 4.1.2 vertieft werden.

3.5 Einordnung der Begriffe Daten, Information und Parameter

Die Datenbeschaffung soll im folgenden als ein Prozess angesehen werden, bei dem Struktur-, Betriebs-, Planungs-, oder Steuerungsdaten so aufbereitet werden, dass sie für Simulationszwecke eingesetzt werden können. Ausgehend von der Durchführung einer Simulationsstudie resultiert hieraus ein mehrstufiger Ablauf, der üblicherweise unter Mitwirkung der betroffenen Fachbereiche durchgeführt wird. Ziel ist es, die verfügbaren und erreichbaren qualitativen und quantitativen Aussagen über das abzubildende System in einer Weise aufzubereiten, dass diese letztlich eine Struktur aufweisen, die direkt in einem Simulationssystem verwendet werden kann (Abb. 22).

3.5.1 Daten

Der erste Schritt ist zunächst im Umfeld des Simulationsexperten zu finden, der anhand einer Aufgabenstellung einen Datenbedarf für das Simulationsprojekt ermittelt. Zu diesem Bedarf muss der Simulationsexperte in einem ersten Schritt das entsprechende Fachwissen in den betroffenen Bereichen lokalisieren, d.h. mit Experten 'Vor-Ort' in Kommunikation treten. Anschließen können diese Experten beginnen, in einem - weitgehend informellen - Prozess die mit der Aufgabe verbundenen Fachsichten sprachlich und inhaltlich abzustimmen. Ist dieser begriffliche Konsolidie-

rungsprozess abgeschlossen, kann mit der eigentlichen Datenbeschaffung begonnen werden.

Hierzu benennt der Vor-Ort Experte zunächst potentielle Datenquellen, von denen *Basisdaten* beschafft werden können. Diese Rohdaten sind dabei so charakterisiert, dass sie für einen anderen Zweck als die Simulation erhoben worden sind - Datenbeschaffenheit und Datenmenge entsprechen deshalb nur in den seltensten Fällen dem Bedarf. Somit ist der zentrale Beitrag des Vor-Ort Experten, mit dem vorhandenen Fachwissen Regeln und Filter zu formulieren, mit denen die Rohdaten so aufbereitet werden können, dass sie für den Simulationsexperten nutzbar werden. Unter Filtern und Regeln sollen in diesem Fall nicht nur die Zugriffsmechanismen auf strukturiert vorliegende Datenbestände verstanden werden (z.B. Datenbanken), sondern allgemeiner jede Art von selektiver Datensammlung, also Zeit-/ Mengenaufnahmen, Interviews usw.

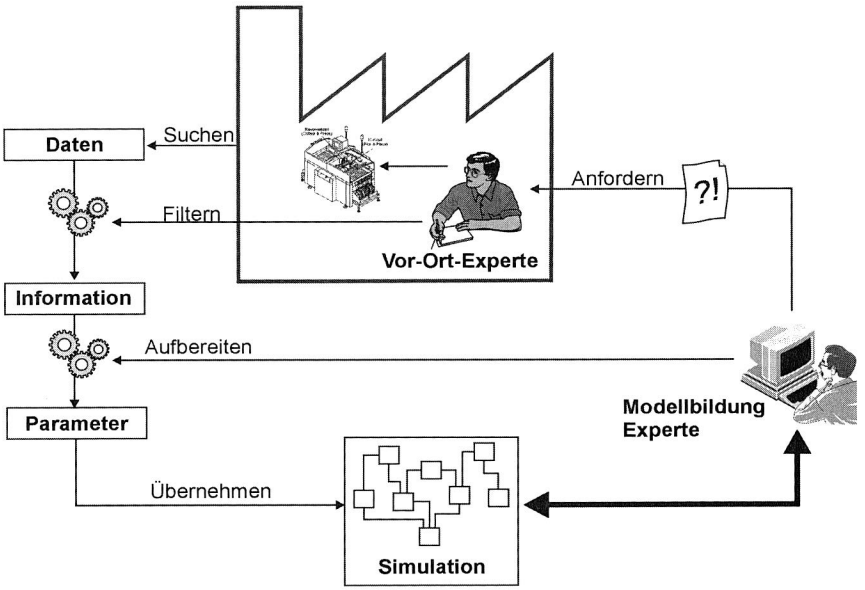


Abb. 22: Begriffbestimmung von "Daten", "Informationen" und "Parametern" im Rahmen der Datenbeschaffung für eine Simulationsstudie

3.5.2 Informationen

Aus Sicht des Simulationsexperten sind durch die Aufbereitung der Rohdaten durch den Vor-Ort Experten *Informationen* über das zu untersuchende System entstanden. Diese Informationen können nunmehr zur Bearbeitung des Simulationsprojekts ge-

nutzt werden. An der Grenze von Daten zu Informationen wechselt somit auch die Verantwortung für die Weiterverarbeitung in den direkten Aufgabenbereich des Simulationsexperten: Die Arbeit mit den Informationen ist nun davon geprägt, diese für das Simulationsprojekt weiter aufzubereiten. Es finden Konsistenzprüfungen, Verdichtungen und weitere vorbereitende Schritte statt, die für den Simulationszweck erforderlich sind.

Diese Grenze ist aus konzeptioneller Sicht ebenfalls interessant, da aus der Sicht eines zu befriedigenden Datenbedarfs hier die Methodengrenze der Simulation und damit der entwerfenden Architektur gesehen werden kann, die mit entsprechenden Schnittstellen einen Zugriff auf die Welt der Daten erhalten muss. Diese Schnittstelle wird in Kap. 4.6.1 beim Entwurf des Wrapper-Konzepts berücksichtigt.

3.5.3 Parameter

Für den Simulationsexperten ist die Bereitstellung von Informationen im weiteren Verlauf des Simulationsprojekts die Grundlage für *Parameter* des Simulationsmodells. Vom Umfang, der Qualität und Verfügbarkeit/Aktualität hängt es ab, ob und wie die bereitgestellten Informationen in das Simulationsmodell übertragen und genutzt werden. Unter Parameter sind daher Informationen zu verstehen, die hinsichtlich Umfang, Beschaffenheit und Format unmittelbar in ein Simulationssystem übernommen werden können.

3.6 Zusammenfassung und weiteres Vorgehen

Die Diskussion über simulationsrelevante Daten belegt, dass die Ablaufsimulation in vielfältigen Problemfeldern eingesetzt werden kann. Allerdings wurde auch deutlich, das besonders die Beschaffung dieser Daten derzeit noch überaus problematisch ist, weil das typische betriebliche Umfeld keine ausreichenden Primärdaten für die Simulation bereitstellt.

Eine abstrakte Bestimmung und Klassifikation der zur Simulation notwendigen Datenarten zeigte, dass zunächst die Struktur, dann die Abläufe und zuletzt die notwendigen Parameter eines Systems zu erfassen sind (Abb. 23). Die Einordnung dieser Schritte motiviert dann zu der Überlegung, die potentiell nutzbaren Primärdaten innerhalb eines Produktionssystems schrittweise auszubauen und anschließend konsequent für Simulationszwecke zu nutzen. Entscheidend war, dass Parameter sowohl als Eingangs- als auch Ergebnisgrößen aufzufassen sind. Ebenso konnte gezeigt werden, dass Strukturen und Abläufe als "Träger" des Datenbedarfs zusätzlich wichtige Metainformationen zur Beschaffung und Verwendung von Daten bereitstellen.

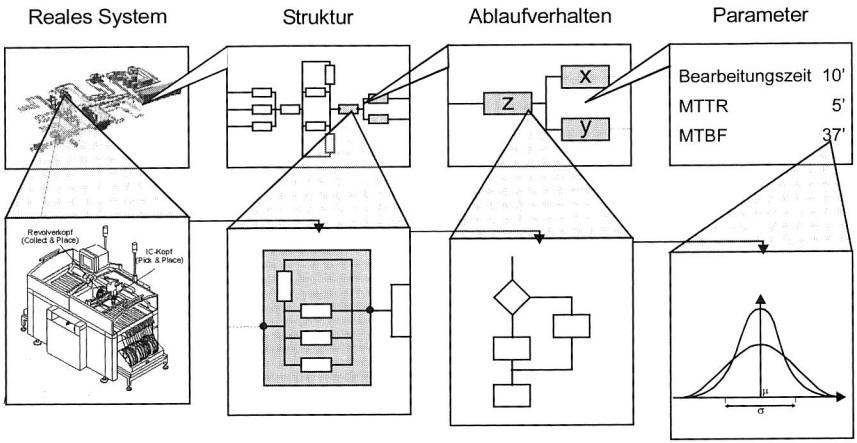


Abb. 23: Zusammenfassende Übersicht der entwickelten Datenklassen

Hieraus lässt sich die weitere Vorgehensweise ableiten: Auf Basis einer noch auszuwählenden Kommunikationsarchitektur soll eine offene Datenkopplung zwischen Anlagen und Simulation konzipiert und realisiert werden, welche die beschriebenen Datenklassen in der Praxis schneller und betriebsnäher nutzbar macht. Die so entstehende Plattform muss anschließend mit einer technisch und ökonomisch tragbaren Integrationsstrategie, einschließlich notwendiger Hilfswerkzeuge, aufgebaut und schließlich unter realistischen Betriebsbedingungen eingesetzt und bewertet werden.

4 Anforderungen an eine offene Datenkopplung zwischen Anlage und Simulation

4.1 Analyse der Anwendungsszenarien

Die vorgehenden Kapitel haben ausgehend vom allgemein anerkannten Simulationsbedarf ein Defizit bei der Beschaffung und Verarbeitung von betriebs- und produktionsbezogenen Daten herausgearbeitet. Aus Anwendersicht wird dieser Mangel typischerweise dadurch deutlich, dass die geforderte Ergebnisqualität nicht erreicht wird oder der hierfür zu treibende Aufwand außerhalb des akzeptablen Zeit- und Kostenrahmens liegen würde. Diese Problematik wird als großes Anwendungshemmnis von potentiellen Nutzern der Simulationüberdurchschnittlich häufig genannt.

Ein Benutzer steht demzufolge während der Realisierungsphase von Simulationsmodellen erstens dem Problem gegenüber, dass aufgrund eines Zusatzaufwands innerhalb der Datenbeschaffung die Modellerstellung nicht schnell genug zum Ziel geführt werden kann. Zweitens wird der langfristig angestrebte Simulationseinsatz durch unzureichende Daten (in Qualität und Quantität) behindert, da die notwendige Initialisierung eines Simulationsmodells eine teilweise erneute Datenbeschaffung erfordert. Damit ist der produktionsprozess-integrierte (online) Simulationseinsatz aufgrund hoher turn-around Zeiten kaum möglich.

Hieraus ergeben sich für den Simulationseinsatz zwei grundsätzliche Anforderungen,

- die schnelle Modellierung bei planungsbegleitenden Studien, einschließlich der Anbindung von Strukturelementen an vorhandene Datenquellen
- sowie kurze Reaktionszeiten bei Datenaktualisierungen während des betriebsbegleitenden Einsatzes,

die durch ein informationslogistisches Architekturkonzept zur Akquisition simulationsrelevanter Daten erfüllt werden müssen.

4.1.1 Planungsbegleitende Anwendung - schnelle Modellierung

Soll Simulation in einem Projekt bereits während einer frühen Planungsphase durchgängig bis zum Serieneinsatz verwendet werden, vollzieht die Modellbildung die Planungsergebnisse anhand einer top-down Vorgehensweise nach [12]. Bei Produktionssystemen werden dabei zunächst grobe Aussagen über die Systemstruktur benötigt, anschließend wird z.B. der Materialfluss innerhalb der einzelnen Fertigungs-

bereiche beschrieben und schließlich werden die Fertigungszellen detailliert berücksichtigt.

Entscheidend für den nachhaltig erfolgreichen Simulationseinsatz ist dabei die Fähigkeit der Simulation, mit den übrigen Planungsergebnissen schritthalten und geeignete Aussagen zum Planungsstand schnell liefern zu können.

Eine schnelle Modellierung setzt dabei eine schnelle Datenbeschaffung voraus. Dies erfordert somit gemäß der in Kap. 3 diskutierten Vorgehensweise zunächst eine strukturelle Identifikation der abzubildenden Systemelemente (Datenklasse S), aus der anschließend Abläufe (Datenklasse A) und Parameter (Datenklasse P) ermittelt werden. Für eine schnellere Modellierung müssen die Attribute der jeweiligen Datenklassen deshalb

- bereitgestellt,
- verwaltet,
- identifiziert und
- übertragen

werden.

Zunächst ist eine Bereitstellung der Attribute auf Seiten der relevanten Datenquellen zu erreichen. Dieser Schritt ist technisch realisierbar, da entweder von einem rechnergestützten Planungsprozess oder von einem Umfeld mit hochautomatisierten Produktionssystemen ausgegangen werden soll. Ebenso werden innerhalb der Planung solcher Systeme vielfach rechnergestützte Werkzeuge verwendet, wie beispielsweise im Zusammenhang mit der methodischen Einbindung der Simulation in das betriebliche Umfeld von [28] umrissen wird. Somit ist für die Bereitstellung der Attribute eine geeignete Systematik zu entwickeln, die einen Zugriff auf solche Informationen innerhalb eines generischen Moduls ermöglicht.

Diese Datenquellen sind dann in einer Weise zu verwalten, dass sie für Simulationszwecke genutzt werden können. Dies bedeutet, dass eine zur Simulationsnutzung bereitgestellte Datenquelle einerseits zwar eine unabhängige Einheit bleiben muss, jedoch andererseits die Informationsgesamtheit zu organisieren ist. Diese Einheit soll sich an den Primärdaten für Simulationszwecke orientieren, was durch geeignete Filter, Regeln und Routinen gewährleistet werden muss.

Die Organisation soll sich dabei an den Bedürfnissen des Simulationsanwenders orientieren, also insbesondere durch geeignete Metainformationen die spätere Identifikation relevanter Datenobjekte durch ein Navigationsmodul ermöglichen: Ein Anwender soll sich mittels des Navigationsmoduls strukturiert in der Informationsge-

samtheit bewegen können. In Bezug auf die schnellere Modellbildung soll damit erreicht werden, dass eine verfügbare Datenquelle schnell gefunden und in das zu erstellende Simulationsmodell als - noch zu konzipierender Baustein - übernommen wird. Dieser Baustein dient dann als Repräsentant der Datenquelle im Simulationsmodell, d.h. durch diesen kann auf alle Informationen und von außen zugängliche Abläufe zugegriffen werden, bzw. alle relevanten Informationen können direkt als Simulationsparameter in das Simulationsmodell übertragen werden.

Alle genannten Komponenten sind durch das Architekturkonzept als Framework (siehe Kap. 6) aufeinander abzustimmen und durch eine Kommunikationsplattform (eine Middleware - Kap. 5) miteinander zu verbinden. Abb. 24 verdeutlicht sowohl dies als auch die Einsatzmöglichkeit bei der Modellierung aus Anwendersicht.

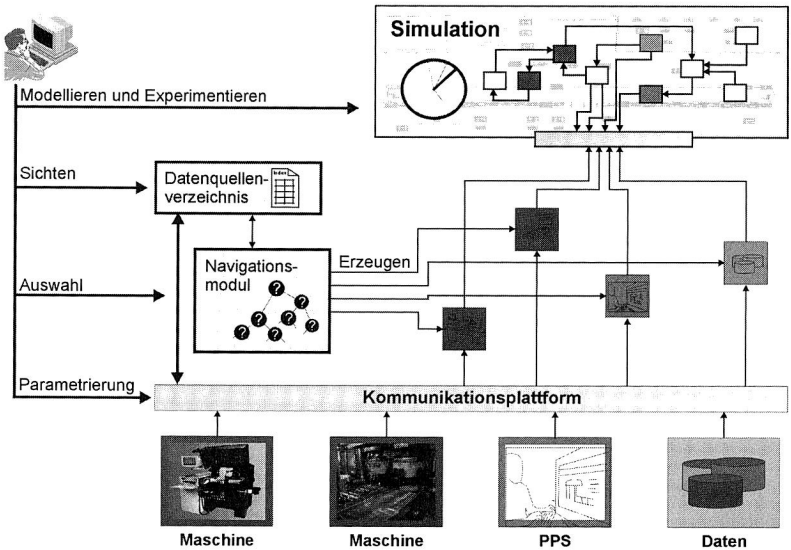


Abb. 24: Konzeptionelle Einsatzmöglichkeiten eines Frameworks bei der Modellierung aus Anwendersicht

Das Potential dieser Vorgehensweise liegt darin, dass

- bei geeigneten Organisationsinformationen direkt simulationsrelevante Strukturinformationen abgeleitet werden können, weil der gewählte Weg während der Navigation bis zur Datenquelle sofort auch die Position innerhalb des abzubildenden Systems aufzeigt, beispielsweise die Position in einer Hierarchie einschließlich Vorgänger- /Nachfolgerbeziehungen.
- der Arbeitsaufwand zur Aufbereitung von Daten reduziert wird, d.h. die Prozesskette Daten → Information → Parameter signifikant verkürzt wird, weil Strukturin-

formationen direkt mit Attributen (Datenklasse S ←→ Datenklasse P) verknüpft sind.

- alle gewonnenen Parameter konsistent und aktuell sind. Dies hat gerade während der Planung den Vorteil, dass eine Parameteränderung als Folge einer Iteration oder Modifikation am Planungsobjekt sofort in das Simulationsmodell übertragen werden kann, weil im Simulationsmodell nicht das Datum selbst, sondern nur eine Referenz auf die Datenquelle hinterlegt und über einen Repräsentanten (Proxy) bereitgestellt wird. Ebenso kann ein Vergleich der Strukturen des Simulationsmodells und des zugrundeliegenden Planungsobjekts weitgehend automatisiert durchgeführt werden, sodass der Simulationsexperte auf eine Diskrepanz zwischen Modell und Realität (aktueller Planungsstand) hingewiesen werden kann.

4.1.2 Betriebsbegleitende Anwendung - kurze Reaktionszeit

Das strategische Ziel beim Einsatz der Simulation ist zunehmend die lebenszeitbegleitende Nutzung der Simulationsmodelle. Hierunter ist speziell die Verwendung der Modelle über die Planungsphase hinaus in der Betriebsphase zu verstehen. Der Grund hierfür ist zunächst in der möglichst langen Nutzung des Simulationsmodells als Investitionsgut zu sehen. Darüber hinaus stellen Simulationsmodelle eine überaus umfassende "Know-how-Konserve" des Planungsobjekts dar, die hinsichtlich des betriebsbegleitenden Einsatzes einen erweiterten Nutzen der bereits getätigten Investition verspricht.

Der Zugriff durch Repräsentanten (Proxys) ist dabei zum betriebsbegleitenden Einsatz gut geeignet. Der transparente (für den Anwender nicht sichtbare) Zugriff auf eine Datenquelle über eine Referenz erfordert hierbei keine weiteren Benutzereingriffe, sodass sich die Möglichkeit zum online-Monitoringeinsatz ergibt: Die Übernahme und Übergabe von Informationen aus der Realität ermöglicht eine Visualisierung anlagen- und situationsspezifischer Parameter und einen Vergleich mit Soll-Eckdaten innerhalb des Simulationsmodells. Im Bedarfsfall, z.B. bei Erkennen einer signifikanten Abweichung der Ist-Situation von der Soll-Situation infolge eines Zulieferengpasses oder eines Maschinenenausfalls, kann innerhalb der Simulation zunächst der Fehlerort visualisiert werden und anschließend können geeignete Bewertungsfunktionen angestoßen werden.

Die online-Kopplung wird dazu zunächst die im Modell enthaltenen Informationen an den aktuellen Zustand der Realität anpassen. Auf der Basis dieser Informationen können dann Funktionen ausgeführt werden, die die Relevanz der Abweichung ermitteln. Dazu wird die Simulation von der Realität 'abgekoppelt' und mit höchstmöglicher Geschwindigkeit wird eine Prognose über die Auswirkungen der jeweiligen Situation erstellt. Der Geschwindigkeitsvorteil der Simulation gegenüber der Realität wird

dazu genutzt, benutzergesteuert oder teilautomatisiert - u.U. mehrere - Handlungsalternativen durchzuspielen, die der auslösenden (signifikanten) Abweichung entgegensteuern und damit die Ist-Situation wieder zurück in den zulässigen Toleranzbereich bringen (Abb. 25).

Der Benutzer soll durch diese Vorgehensweise dabei unterstützt werden, gezielt und nachhaltig auf Störungen oder sonstige signifikante Abweichungen zu reagieren, d.h. das vorgestellte Vorgehen stellt einen wichtigen Bestandteil eines operativen Assistenzsystems dar, welches beispielsweise ergänzend zu mittelfristig und langfristig angelegten Assistenzsystemen genutzt werden kann, wie sie etwa in [93,94] beschrieben werden.

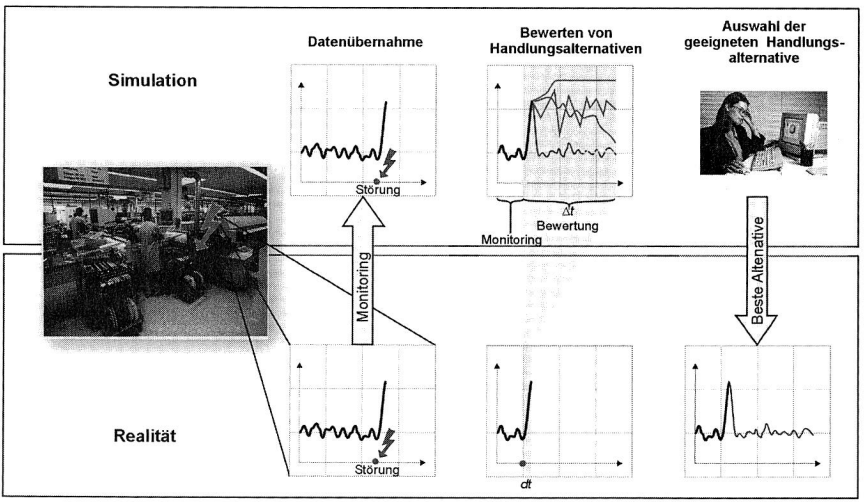


Abb. 25: Konzeptionelle Einsatzmöglichkeiten in der betriebsbegleitenden Anwendung

4.2 Anforderungen an das System

Aus den genannten Anwendungsszenarien ergeben sich eine Reihe von Anforderungen an das System. Der wichtigste Punkt ist dabei der sichere Umgang mit einer erheblichen Datenmenge. Die einzelnen Elemente dieser Menge sind erstens innerhalb des Produktionssystems verteilt, d.h. jede vorhandene Datenquelle muss schnell lokalisiert werden. Zweitens muss überlegt werden, wie mit unterschiedlichen Datenverfügbarkeiten umgegangen werden muss und drittens ist für den langfristig angelegten Einsatz die Skalierbarkeit des Systems entscheidend, um auf Erweite-

rungen/Veränderungen des Planungsobjekts oder realen Systems flexibel reagieren zu können.

4.2.1 Datensicherheit

Kaum einem anderen Bereich werden solche Steigerungsraten vorausgesagt wie dem internetbasierten Informationsaustausch (z.B. eCommerce). Ob die weitere Entwicklung die hochgesteckten Erwartungen erfüllen wird, hängt entscheidend von der Zuverlässigkeit und Sicherheit der darauf aufbauenden Anwendungen ab.

Eines der Grundprobleme dabei ist der sichere Informationsaustausch in einem weitgehend offenen Netzverbund, der ursprünglich nicht zur Übertragung sicherheitskritischer Informationen entwickelt worden ist. Daher sollten sensitive Daten nicht ohne zusätzliche Sicherheitsmaßnahmen übertragen werden. Datenschutz und Datensicherheit sind demzufolge an technische und organisatorische Voraussetzungen geknüpft [95].

Die im Architekturkonzept übermittelten Informationen sind zum Grossteil sicherheitskritisch, da planungs- und produktionsrelevante Daten in erheblichem Umfang und mit signifikanter Aktualität verfügbar gemacht werden. Diese Situation verschärft sich durch die Tendenz, dass Unternehmen sich zu weltweiten 'virtuellen' Produktionsverbünden zusammenschließen. Die einfache Abschottung in einem Intranet wäre daher ein Anwendungshemmnis. Daher sind geeignete Maßnahmen zum sicheren Datenaustausch in einem wandlungsfähigen, dezentralen Extranet zu ergreifen. Zuletzt ist darauf zu achten, dass es sich hierbei nicht nur um einem vordergründigen Schutz handelt darf. Beispielsweise zeigt die aktuelle Diskussion im Zusammenhang mit ECHELON [96] im Europarat, dass das organisierte Ausspähen wirtschaftlich relevanter Daten eine ernsthafte Standortgefährdung darstellt.

Die Architektur hat deshalb als Basisfunktionalität einen sicheren Informationsaustausch zu gewährleisten, d.h. es sind gesicherte Übertragungskanäle anzubieten, die für den Benutzer transparent sind und technisch mindestens die folgenden Eigenschaften aufweisen:

Schützenswerte Informationen sollen nur (symmetrisch oder asymmetrisch) verschlüsselt übertragen und zusätzlich signiert werden. Während die Verschlüsselung einen Schutz gegen den Verlust der Vertraulichkeit darstellt, wird durch die Signatur einer Datei eine unerkannte Änderung an dieser Datei verhindert, wodurch ein Schutz gegen Integritätsverletzungen aufgebaut wird.

Für eine Übertragung der Daten zwischen Personen ist eine Verschlüsselung und Signatur direkt auf Anwendungsebene sinnvoll. Soll die gesamte Kommunikation

zwischen Rechnern oder Netzen gesichert werden, ist eine Verschlüsselung auf einer tieferen Ebene (z.B. Betriebssystem) sinnvoll, um z.B. eine gesicherte Verbindung zwischen Sender und Empfänger zu ermöglichen. Beim symmetrischen Verfahren müssen zudem die benutzten Schlüssel auf einem vertrauenswürdigen Weg zwischen den beteiligten Partnern ausgetauscht werden.

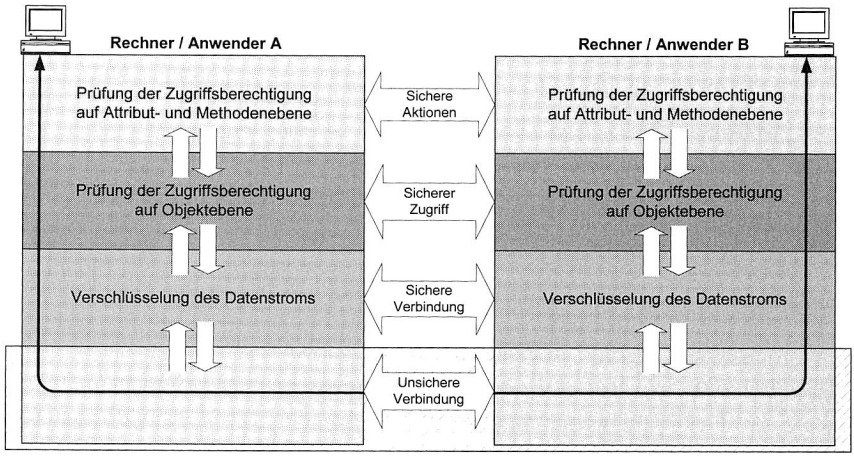


Abb. 26: Gesicherte Datenübertragung durch ein Dreistufenkonzept: Sichere Verbindungen, Zugriffe und Aktionen

Insgesamt sind drei Sicherheitsstufen zu berücksichtigen (Abb. 26):

1. Das Basiskommunikationssystem muss nach außen sicher sein. Es muss also gewährleistet werden, dass ein Außenstehender keine Kenntnis über die tatsächlich übertragenen Informationen erhält. Die Kommunikation sollte daher grundsätzlich auf der niedrigst möglichen Ebene bereits sicher sein. Ein Beispiel hierfür wäre etwa SSL (secure socket layer) [97].
2. Innerhalb der Architektur ist, ergänzend zu Punkt 1, die Vertrauenswürdigkeit bei der Kommunikation zwischen den Datenobjekten zu gewährleisten, etwa indem sich Objekte gegenseitig als "Berechtigt" ausweisen.
3. Zuletzt kann innerhalb der Objekte für jede Methode und jedes Attribut eine gestufte Berechtigung eingesetzt werden, etwa um das Lesen, Schreiben, Ändern oder Ausführen durch unterschiedliche Benutzerkreise kontrollieren zu können.

Speziell die erste Forderung wird in den folgenden Kapiteln aufgegriffen und bei der Auswahl der Middleware berücksichtigt. Die letzten beiden Forderungen fliessen konzeptionell in Kap. 6 ein.

4.2.2 Benutzertransparente Datenlokalisierung

Die Datenlokalisierung kann als Prozess aufgefasst werden, bei dem der Anwender die im Datenbestand verfügbaren Informationen strukturiert durchsuchen kann und damit schnell zu der/den relevanten Datenquelle(n) innerhalb des Datenbestands gelangt.

Im einzelnen sind hierzu folgende Module und Funktionen vorzusehen:

- **Registrierung von Datenobjekten**

Die technische Voraussetzung der benutzertransparenten Datenlokalisierung ist, dass sich Datenquellen in einer dazu notwendigen Verwaltungsinstanz registrieren. Dabei muss nicht nur ihre Verfügbarkeit bekannt gegeben werden, sondern jede Datenquelle muss zusätzlich Informationen über sich selbst geben (Metadaten), z.B. räumliche Position, Anordnung im Prozess sowie das Vorhandensein von technischen Daten und die Bereitstellung von Funktionen aus der datenquelleneigenen Steuerung.

Die Registrierung ist daher ein von der Datenquelle aktiv gesteuerter Ablauf. Über den Zeitpunkt, zu dem die Registrierung (oder auch das Abmelden aus dem Datenbestand) durchgeführt wird, entscheidet ausschließlich die Implementation der Datenquelle. Ebenso entscheidet die Datenquelle selbst darüber, welche Informationen bereitgestellt werden.

Die bei der Registrierung der Datenquelle verwendeten Informationen sollen sich an der Datenklasse S (Name, Standort, Relationen zu anderen Elementen) orientieren. Die bereitgestellten Informationen sollen eine inhaltliche Entsprechung in den Datenklassen P und A haben, etwa durch ausführliche Klartextbezeichnungen. Hierdurch ist es dem Anwender möglich, das registrierte Objekt einfach anzusprechen.

Die notwendige Transformation in eine architektureigene Bezeichnung erfolgt dann mittels eines Datenquellenverzeichnisses. Die Transformation selbst ist dabei unabhängig von räumlichen Anordnungen o.ä. und gewährleistet somit einen systemunabhängigen, abstrakten Zugriff auf die Datenquelle (Name→ Verzeichnis→ Datenquelle).

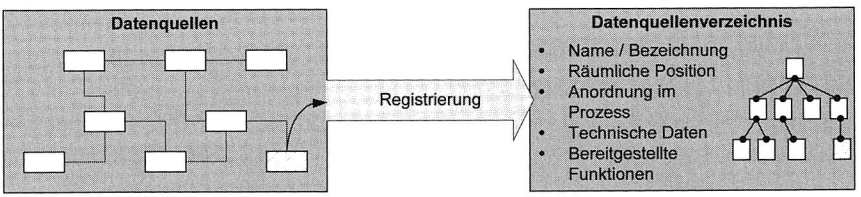


Abb. 27: Vorgehensweise bei der Registrierung von neuen Datenobjekten

- **Verwalten von Datenobjektreferenzen in Hierarchien**

Die Datenquellenobjekte werden zunächst als geschlossene und unabhängige Einheiten betrachtet. Es kann nicht davon ausgegangen werden, dass sie außer den bereitgestellten Daten auch über Informationen verfügen, welche Auskunft über den dynamischen Kontext geben, in den sie eingebettet sind - die Datenquelle registriert sich und überlässt die Organisation der Zusammenhänge der Verwaltungsebene.

Wird die Datenquelle aktiviert, so ermittelt diese zunächst ihre eigene Datenquellenreferenz, welche als eindeutige und unverwechselbare Kennung aufgefasst wird. Bei der Aktivierung sind Parameter zu übergeben, die festlegen, welche Position die Datenquelle innerhalb verschiedener Sichten auf das Produktionssystem einnimmt. Diese ist, in Anlehnung an die Ergebnisse der VDI-Arbeitskreises "Hierarchische Simulationsmodelle" die Position innerhalb der Hierarchisierungskriterien Produkt, Ressource und Prozess [98].

Unterschiedliche Sichten werden gewählt, um dem Anwender eine möglichst problemgerechte, einfache und umfassende Nutzung der Informationsbestände zu erlauben. Dies entspricht besonders der erste Kernforderung nach einer planungsbegleitenden Anwendung [99]. Durch die Hierarchisierung nach unterschiedlichen Kriterien können die vorhandenen Informationsbestände zielgerichtet nach relevanten Datenquellen durchsucht und anschließend in der Simulation genutzt werden.

Beim *Produkt* ist eine zeitliche Sichtweise möglich, z.B. der Produktlebenszyklus, bei dem über die Phasen Marketing, Produktplanung, Produktentwicklung, etc. ein zunehmender Erkenntnisgewinn zu verzeichnen ist. Hierbei soll ebenso eine statische Teilebeziehung hergestellt werden, die etwa über eine Stückliste oder ein CAD-Format repräsentiert wird, wie auch eine funktionale Hierarchie oder eine dynamische Teilebeziehung. Zusätzlich kann die Sichtweise auf ein Produkt auch einer Organisationshierarchie entsprechen, wobei z.B. Waren- und Teilegruppen zusammengefasst werden können.

Bei der *Ressource* ist - ähnlich wie beim Produkt - der Lebenszyklus des Produktionsmittels ebenfalls ein hierarchisierendes Kriterium. Ebenso kann die "Struktur" des Produktionsmittels den physikalischen Aufbau kennzeichnen, z.B. Greifzange ist Teil des Greifers, Greifer ist Teil des Handhabungssystems, Handhabungssystem ist Teil der Maschine, Maschine ist Teil der Autonomen Zelle usw. Genauso ist eine funktionale Sichtweise auf Ressourcen sinnvoll, bei denen Eigenschaften definiert werden, z.B. "kann bohren", "kann fräsen", "transportiert" usw. Bei der Ressource kann ebenfalls eine "Organisationshierarchie" angenommen werden, die die aufbauorganisatorische Betrachtung heranzieht. Dies ist meist eine "gehört zu" - Beziehung.

Ein *Prozess* stellt eine Verbindung zwischen einem Produkt und einer Ressource dar und unterliegt selbst somit keinem Hierarchisierungskriterium. Vielmehr entsteht die Hierarchisierung implizit durch die Relation, die ein Knoten "Prozess" innerhalb einer Hierarchie zwischen einem Produkt und einer Ressource herstellt.

Jedes hierarchisierende Kriterium bildet einzeln einen Baum, dessen Informationsumfang mit zunehmender Tiefe zunimmt. Um verschiedene Fachsichten repräsentieren zu können, kann jede Datenquelle in unterschiedlichen Hierarchiebäumen registriert sein. Will der Benutzer während der Navigation den Baum (das Kriterium) wechseln, besteht die Gefahr eines Zyklusses. Dies wird dadurch vermieden, dass bei der Nutzung der Datenquellenverzeichnisse der Weg durch einen oder mehrere Bäume aufgezeichnet wird (Navigationsmarken). Damit kann ein möglicher Zyklus erkannt werden und der Anwender hat die Möglichkeit, den Pfad bis zum Blatt eines Baumes (eine Datenquelle) zurückzuverfolgen.

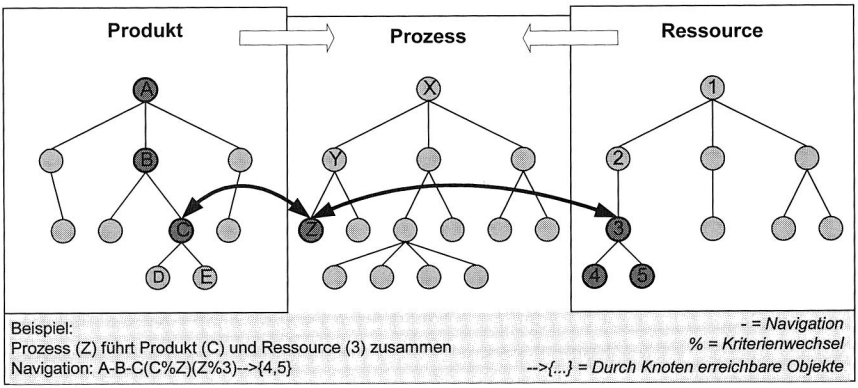


Abb. 28: Beispiel der Navigation durch die hierarchiebestimmenden Kriterien "Produkt", "Prozess" und "Ressource"

Die Hierarchisierung bietet dem Anwender eine Systematik, um sich in den simulationsrelevanten Datenbeständen eines komplexen Produktionssystems innerhalb filternder Fachsichten zu bewegen und damit simulationsrelevante Datenquellen schnell zu identifizieren.

Eine identifizierte Datenquelle kann schließlich als "Objekt" in ein Simulationsmodell als Element der Datenklasse P, A oder S übernommen werden, ohne dass im Idealfall eine weitere manuelle Aufbereitung der gefundenen Informationen notwendig wäre. Der Pfad entlang der Navigationsmarken gibt zusätzlich eine Auskunft darüber, wie diese Datenquelle gefunden wurde. Dieses Vorgehen soll als *datengetriebene Modellierung* bezeichnet werden.

- **Konsistenzprüfung des Datenbestands**

Neben der Verwaltung und Zuordnung von Objektnamen zu Objektreferenzen aller registrierten Datenquellen hat die Verwaltungsinstanz als Aufgabe weiterhin eine permanente Konsistenzprüfung durchzuführen.

Hierbei ist zu prüfen, ob alle registrierten Objekte verfügbar sind, tatsächlich die gemeldeten Informationen bereitstellen und in ihrer Gesamtheit korrekt in der Hierarchie angeordnet sind. Insbesondere der letzte Punkt ist erforderlich, da ein Objekt innerhalb der Hierarchie mehrfach erscheinen kann und der Wechsel zwischen den Hierarchisierungskriterien bei falschen Verweisen den gesamten Informationsbestand für den Anwender nutzlos werden lässt.

Zu prüfen ist an dieser Stelle, ob die Verweise zwischen den Navigationsmarken vollständige Vorwärts- und Rückwärtsreferenzen enthalten. Diese können ggf. über die gemeinsame Datenquellereferenz wieder hergestellt werden.

Genauso ist zu prüfen, ob als verwendet markierte Objekte tatsächlich in Benutzung sind, oder ob in diesem Zusammenhang belegte Systemressourcen freigegeben werden können. Dies ist aus technischen Gründen notwendig, da für die Verwaltung der Datenquellengesamtheit Zusatzinformationen erstellen werden, die nur während der tatsächlichen Nutzung Systemressourcen belegen sollen.

4.2.3 Sichern von Informationen aus Datenobjekten

Hinsichtlich der Robustheit des Datenakquisitionsprozesses reicht es nicht aus, wenn ausschließlich Informationen von Datenquellen des Produktionsprozesses beschafft werden können. Ist die Kommunikation zur Datenquelle unterbrochen oder werden Daten aus der Vergangenheit ('historisierte' Daten) verlangt, ist es sinnvoll, auf früher gespeicherte Daten zugreifen zu können. Deshalb sollen alle relevanten Informationen einer Datenquelle bedarfsweise für einen gewissen Zeitraum gespeichert werden. Dies soll entweder zyklisch (in gewissen Zeitabständen, etwa für die Historisierung), simulationssynchron (im Kontext eines planungs- oder betriebsbegleitenden Vorgehens) oder ereignisgesteuert durch die Datenquelle selbst (simulationsasynchron) erfolgen, z.B. als Reaktion auf eine definierte Veränderung im Systemzustand (Abb. 29). Zu unterscheiden ist hierbei zwischen angefallen Daten während der Planung, z.B. einer Plandatenbasis [28], und Betriebsdaten, die u.U. aus Systemen der BDE oder aus anderen Datenquellen (z.B. Steuerungsrechner) ermittelt werden können.

4.2.4 Robustheit des Akquisitionsprozesses

Für die Akquisition ist eine robuste Kommunikation zwischen Simulation und Anlagenkomponenten und anderen Datenquellen sicherzustellen: Bei der Anbindung realer Anlagenobjekte kann eine Störung der Kommunikation zur Unterbrechung einer laufenden Simulation und evtl. zum Verlust der bisherigen Rechenresultate führen. Deshalb soll die Kopplung fehlertolerant ausgelegt werden, so dass eine automatische Umschaltung auf eine Datenbank als alternative Datenquelle zum realen Anlagenobjekt vorgenommen werden kann. Um dabei insgesamt die Aktualität der Daten in der Datenbank zu gewährleisten, werden die relevanten Parameter aus dem Prozess darüber in einer back-up Datenbank gespeichert.

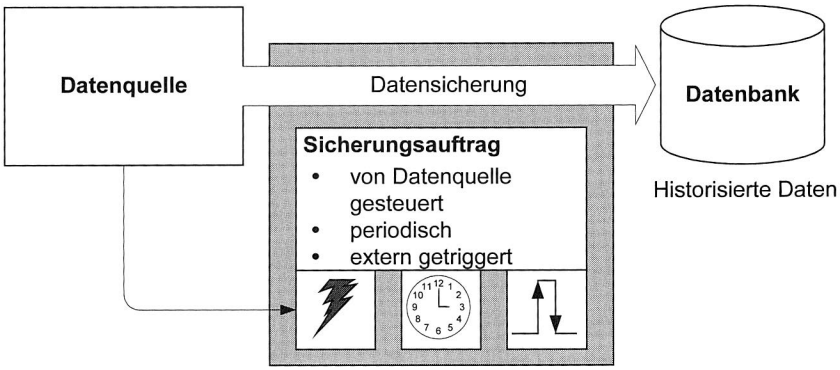


Abb. 29: Datenquellengesteuerte, periodische oder extern getriggerte Sicherung von Informationen aus Datenquellen zur Erzeugung historisierter Daten

Die eigentliche Umschaltung soll durch benutzerdefinierte Vorgaben von Strategien erfolgen. Diese prüfen zunächst, ob eine angeforderte Datenquelle verfügbar ist. Sollte dies zum betreffenden Zeitpunkt, z.B. aufgrund einer Kommunikationsstörung, nicht der Fall sein, kann eine Liste alternativer Datenquellen hinterlegt werden, zu denen dann eine Verbindung aufgebaut werden soll. Mittels solcher "Präferenzenlisten" können damit einer Modellkomponente verschiedene Datenquellen zur Initialisierung des Modells und zur Parametrierung zugeordnet werden. Bei gestörter Kommunikation zu einem realen Anlagenobjekt wird dann beispielsweise auf eine alternative back-up Datenquelle umgeschaltet, ohne dass der eigentliche Datenbeschaffungsprozess unterbrochen würde. Entsprechend muss die verwaltende Schicht ein Attribut "Herkunft" bereitstellen, um bedarfsweise unterscheiden zu können, ob ein Alternativdatum bereitgestellt wurde.

Als Teilanforderung an das Gesamtsystem ergeben sich hinsichtlich der Robustheit somit zwei Arbeitspakete:

- Einrichten einer zeit- und/oder ereignisgetriggerten Sicherung von Prozessdaten als Erweiterung der Datenquellenkopplungen zur robusten Betriebsdatenerfassung. Diese Aktivitäten müssen unabhängig von den Prozessen der Datenbeschaffung für einen Simulationsnutzer, d.h. im Hintergrund, ablaufen. Hierzu werden die Mechanismen zur Erzeugung historisierter Daten (Kap. 4.2.3) genutzt.
- Umsetzen einer regelbasierten Präferenzenliste für einen zustandsabhängigen Zugriff auf gleichartige Datenquellen. Entsprechende Regeln werden entweder als IF...THEN...ELSE formuliert oder nutzen funktionsorientierte oder regelbasierte Beschreibungsmodelle [100]. Mit diesen wird dann im Bedarfsfall eine Alternativdatenquelle gesucht oder 'abgeleitet'.

4.2.5 Erweiterbarkeit und Offenheit bezüglich vorhandener Systeme

Die Erweiterbarkeit des Systems ergibt sich als zwingende Voraussetzung, um das System im betrieblichem Umfeld einsetzen zu können. Erweiterbarkeit bedeutet daher, dass das System einfach um zusätzliche Parameter und Datenstrukturen ergänzt werden kann, sodass diese innerhalb des Systems verwaltet und bereitgestellt werden können. Konzeptionell stellt sich diese Anforderung als der Entwurf einer allgemeinen Datenklasse (einer Generalisierung im Sinne von Rumbough [101]) dar, die durch weitere Informationen zu einer Datenklasse (Datenquelle) für den jeweiligen Zweck genauer spezifiziert werden kann. Diese Datenklasse wird als Basis verwendet und gemäß des Einsatzzweckes erweitert.

Die 'allgemeine Datenklasse' soll damit grundsätzlich den Implementationskern für alle Erweiterungen des Systems bilden, d.h. alle spezifizierenden Klassen sollen in einer "is-a" Relation zu dieser stehen. Beispiel: Ist bei einem Bestücker vom Typ A der Yield (Ausbringungsanteil an Gutteilen) von Interesse, wird zunächst begonnen, den Bestücker als Datenquelle aufzufassen, d.h. es wird die 'allgemeine' Datenklasse verwendet. Diese wird anschließend um den bestückerspezifischen Parameter 'Yield' erweitert. Dies hat zur Folge, dass die daraus resultierende Klasse 'Bestücker A' eine Spezialisierung der 'allgemeinen' Datenklasse ist. Existiert nun noch eine zweite Klasse 'Bestücker B' mit einem anderen Parameter, so sind beide Klassen auf der Ebene der Spezialisierung zwar unterschiedlich, habe jedoch immer noch einen gemeinsamen (allgemeinen) Kern. Dieser kann universell genutzt werden, um beispielsweise die spezialisierenden Charakteristika zu ermitteln.

Weiterhin ist festzulegen, wie neue Datenquellen in das System eingebunden werden können. Dies betrifft sowohl die Verwaltung solcher Datenquellen als auch den tatsächliche Zugriff auf die relevanten Daten: Der Zugriff setzt also ein Konzept voraus, welches aus Architektursicht die Datenquelle uniform kapselt und über einheitli-

che Schnittstellen den Zugriff auf unterschiedliche Systeme mit objektorientierten Beschreibungs- und Implementierungsmethoden ermöglicht.

Als funktionaler Kern wird zu diesem Zweck das Entwurfsmuster der Wrapper nach [102] verwendet (Abb. 30). Wrapper stellen eine Kapsel um ein bestehendes (Teil-)System dar, die dessen Integration in ein Umfeld erlaubt, für welches das gekapselte System ursprünglich nicht geplant war. Dabei kommunizieren Wrapper über systemspezifische Schnittstellen mit diesem gekapselten System und bieten dessen Funktionalität, bzw. wesentliche Teile hiervon, über ihre externen Schnittstellen anderen System (Klienten) der Architektur an.

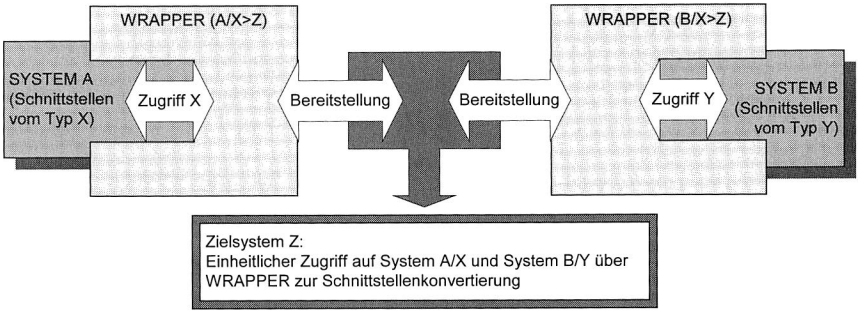


Abb. 30: Das Wrapper-Konzept

Wichtig dabei ist, dass ein Wrapper zwar extern sehr unterschiedliche Funktionalitäten der Datenquelle verwalten muss (Abb. 30 - *Zugriff*), diese jedoch innerhalb der Architektur und in bezug auf die zu kapselnde Kategorie identische Schnittstellen aufweisen muss (Abb. 30 - *Bereitstellung*). Dies ist eine funktionale Forderung an die 'allgemeine' Datenklasse.

Weiterhin kann durch die einheitliche Verwendung der Wrapper einfach ein logischer Zusammenhang zu den Repräsentanten im Simulationsmodell hergestellt werden, da - abgesehen von der noch zu beschreibenden Verwaltungsschicht - simulationsseitig die Repräsentanten und datenquellenseitig die Wrapperobjekte den gleichen Datenbestand enthalten.

4.3 Unterteilung der Architektur in 3 Schichten

Nachdem die Anforderungen an die zu konzipierende Architektur aus Anwendersicht definiert wurden, erfolgt die Zerlegung des Gesamtsystems in kooperierende funktionale Schichten, auf deren Basis die Realisierung des Gesamtsystems erfolgt. Ein derartiges Schichtenmodell darf nicht mit den Schichten des *ISO*-Referenzmodells für Kommunikationssysteme verwechselt werden, da diese nur logische Bezugsebe-

nen identifizieren. Im Gegensatz dazu unterscheiden Schichten eines objektorientierten Client/Server-Systems zwischen den Komponenten einer Architektur [103].

Bei der Betrachtung von verteilten objektorientierten Systemen erfolgt oft die Einteilung in drei Schichten (Abb. 31).

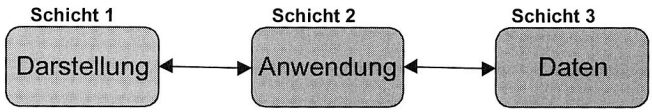


Abb. 31: Die objektorientierte 3-Schichten Architektur

In der ersten Schicht befinden sich diejenigen Komponenten, die auf die angebotenen Dienste und Daten zugreifen wollen. Sie wird auch als Darstellungsschicht bezeichnet, da hier die eigentliche Interpretation und Verarbeitung der in der Architektur ausgetauschten Informationen erfolgt.

Die mittlere Schicht wird als Koppelmedium zwischen der ersten und der dritten Schicht angesehen. Über die Schnittstellen der in ihr enthaltenen Komponenten können Klienten aus der Darstellungsschicht die Daten und Dienste der dritten Schicht benutzen. Sie stellt aus der Sicht eines Darstellungsschicht-Klienten die eigentliche Anwendung dar, da die Bereitstellung aller Dienste und Daten über sie erfolgt. Aus diesem Grund wird sie auch als Anwendungsschicht bezeichnet.

In der dritten Schicht befinden sich diejenigen Architekturkomponenten, die als eigentliche Quelle der Informationen und Dienste angesehen werden; dies sind z.B. Datenbanken, Server-Systeme usw. Durch sie erfolgt die Realisierung der Dienste, die der Darstellungsschicht über die Anwendungsschicht angeboten werden. Sie wird als Datenschicht bezeichnet.

Beim näherem Betrachten der Anforderungen an die Architektur und der in ihr agierenden Komponenten lassen sich die gleichen Strukturen ausmachen; auch hier kann das System in eine Darstellungsschicht, eine Anwendungsschicht und eine Datenschicht unterteilt werden (Abb. 32).

In der Darstellungsschicht befinden sich das eingesetzte Simulationswerkzeug und die für dessen Integration benötigten Objekte. Die Simulation ist der Klient der Architektur, der auf die Dienste und Daten der Schicht-3-Systeme zugreifen will. Die von der Simulation angeforderten Daten beziehen sich i.d.R. auf den von ihr simulierten Fertigungsprozess und werden von ihr benötigt, um ein möglichst genaues Wissen über den aktuellen Prozesszustand zu erlangen (Datenakquisition).

In der Anwendungsschicht befinden sich Objekte, von denen die Dienste und Daten der Datenschicht unsichtbar für den Anwender (transparent) den Klienten der Darstellungsschicht angeboten werden. Sie akquirieren die benötigten Informationen aus der Datenschicht und repräsentieren diese auf eine einheitliche Art. Dadurch können Klienten auf die Dienste und Daten mehrerer Datenschicht-Systeme zugreifen, ohne Details über deren Aufbau zu wissen.

Der Nutzen der gewählten Dreischichtenarchitektur wird am Beispiel der bereits beschriebenen Präferenzenlisten noch deutlicher: Die Umschaltung aufgrund der Präferenzenliste soll für den Benutzer (in der Darstellungsschicht) transparent erfolgen. Diese kann daher nur in der Anwendungsschicht vorgenommen werden, da ein direkter Zugriff auf die Datenschicht die Kommunikation bereits unterbrochen hätte.

In der Anwendungsschicht agierende Objekte lassen sich abhängig von den Aufgaben, die ihnen zufallen, in drei Kategorien einteilen:

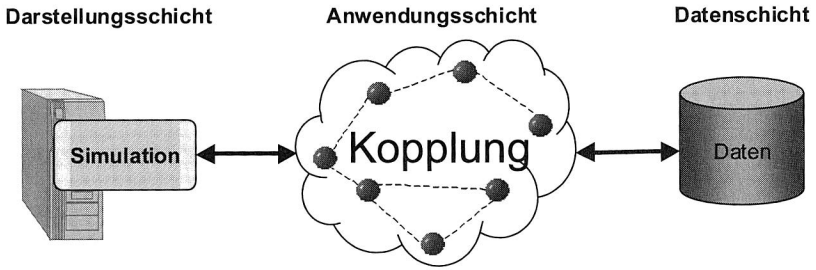


Abb. 32: Schichten bei der Kopplung von Datenquellen und Simulation

- Zur ersten Kategorie gehören Objekte, mit denen die Systeme der dritten Schicht in die Anwendungsschicht integriert werden. Sie ermöglichen den Komponenten der Architektur den Zugriff auf die Systeme der Datenschicht.
- Zur zweiten Kategorie gehören Objekte, mit denen die Objekte der Anwendungsschicht auf mehrere Objekte der 1. Kategorie zugreifen können. Solche Objekte können auch mit Relaisstationen (engl. *Repeater*) verglichen werden, da sie die benötigten Daten aus mehreren Datenschicht-Systemen akquirieren, die beschafften Informationen zwischenspeichern, und diese, ohne sie zu verändern, der Simulation anbieten.
- Zur dritten Kategorie gehören Objekte, deren Aufgabe die Repräsentation von Strukturinformationen realer Produktionssysteme ist. Basierend auf diesen Organisationsstrukturen erfolgt die Generierung von Modellen, die in einem Simulator ablaufen können.

In der Datenschicht befinden sich PPS-Systeme, Datenbanken, BDE-Systeme sowie die Steuerungen von realen Fertigungskomponenten. Sie alle werden als Server angesehen, die Daten oder Dienste bereitstellen, die von einem Simulator genutzt werden können.

4.4 Betrachtung der Datenschicht (Datenquellen)

Bei den Komponenten, die zur Datenschicht gerechnet werden, handelt es sich um alle Systeme eines Planungsobjekts oder Produktionssystems, die über simulationsrelevante Informationen verfügen.

Die Kommunikation mit diesen Systemen erfolgt in der Regel über herstellerspezifische Schnittstellen, die meistens auf üblichen Kommunikationsdiensten wie etwa RPC's, shared Memory, TCP/IP etc. basieren (Abb. 33). In diesen Systemen gespeicherte Daten können durch die obigen Schnittstellen und über entsprechende Objekte innerhalb der Anwendungsschicht verfügbar gemacht werden. Von hier aus sind sie dann auch für die Simulation zugänglich.

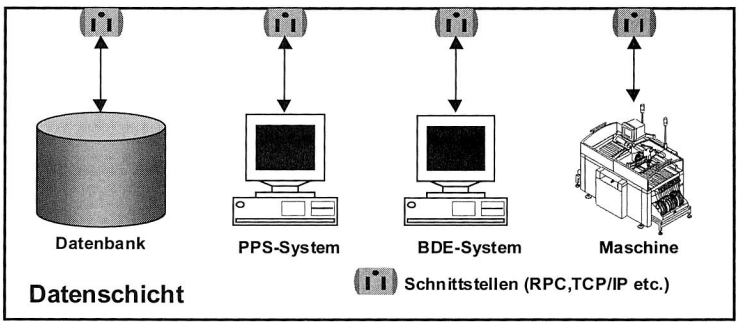


Abb. 33: Die Systeme der Datenschicht

Voraussetzung innerhalb der Datenschicht ist die Fähigkeit der Architektur, die Objekteigenschaften (Attribute und Funktionalitäten) intern in einer vereinheitlichten Form anbieten zu können und damit für die Anwendungsschicht nutzbar zu machen. Dies wird durch das Wrapper-Konzept ermöglicht. Hierzu ist während der Realisierung ein Kompromiss zu finden, der einerseits die dazu notwendige Flexibilität beim Zugriff und andererseits die gebotene Integrierbarkeit aufeinander abstimmt. In diese Überlegungen müssen weiterhin die sehr unterschiedlichen Datenquellen bei Planung und Betrieb unter Berücksichtigung der vielfältigen Simulationsziele einbezogen werden.

Von den Wrappern müssen drei unterschiedliche 'generische' Informationstypen bereitgestellt und verwaltet werden: Informationen und Parameter, Funktionen sowie Ereignisse.

Informationen und Parameter sind zunächst im Rahmen des Wrappings bereitzustellen. Da diese Datenmengen schnell einen erheblichen Umfang annehmen werden, müssen weiterhin die durch jede einzelne Datenquelle bereitgestellten Informationen und Parameter in einem Verzeichnis abgelegt werden, das dem Nutzer schnell eine Übersicht über den insgesamt verfügbaren Datenbestand gibt. Dieses Verzeichnis ist auf unterster Ebene ein notwendiger Bestandteil der Navigation im gesamten Informationsbestand.

Funktionen einzelner Datenquellen sollen genutzt werden, um im Sinne einer top-down Modellierung eine Blackbox im Simulationsmodell nicht weiter auflösen zu müssen. Vielmehr wird statt der abstrahierten Ablaufbeschreibung im Simulationsmodell ein Repräsentant verwendet, der - soweit möglich - die datenquelleninterne Ablaufsteuerung nutzt. Beispielsweise verfügen Bestücksysteme über eigene Steuerungen, auf die über standardisierte Schnittstellen zugegriffen werden kann (GEM/SECS II oder OPC). Statt nun im Simulationsmodell das Verhalten zu beschreiben, kann ein Wrapper eingesetzt werden, der Funktionsaufrufe aus der Simulation direkt in Steuerungsanfragen des Bestückers umsetzt und die Ergebnisse (etwa die Bearbeitungsdauer) an die Simulation zurückliefert. Dementsprechend ist - bei höherer Beschreibungsgenauigkeit - simulationsseitig keine detailliertere Beschreibung mehr notwendig.

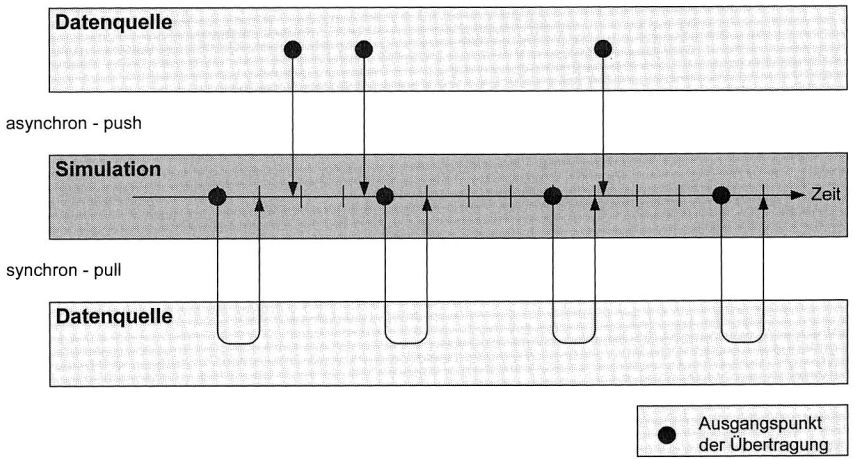


Abb. 34: Betriebsarten der Datenschicht: synchron zur Simulationslaufzeit (pull - unten) und asynchron zur Simulationslaufzeit (push - oben)

Weiterhin ist neben der synchronen Akquisition von Informationen (Abfragen, die synchron zur Simulationslaufzeit initiiert werden - pull-Betrieb) auch vorzusehen, dass eine Datenquelle selbstständig Ereignisse, z.B. Störungen, asynchron zu einem Simulationslauf meldet (push-Betrieb) (Abb. 34). Diese Möglichkeit stellt einen Schlüsselfaktor für die technische Realisierung des online-Monitorings dar, weil hierdurch wichtige Informationen aktiv zur Simulation geleitet werden können.

4.5 Betrachtung der Darstellungsschicht (Simulation)

Bei der Einteilung der Architektur in drei Schichten wurde festgelegt, dass sich in der Darstellungsschicht die eingesetzten (Simulations-) Werkzeuge befinden. Ihre hauptsächliche Rolle wurde als die eines Klienten definiert. Es werden Funktionalitäten der Anwendungsschicht (s.u.) genutzt, um an die Informationen der Datenschicht zu gelangen.

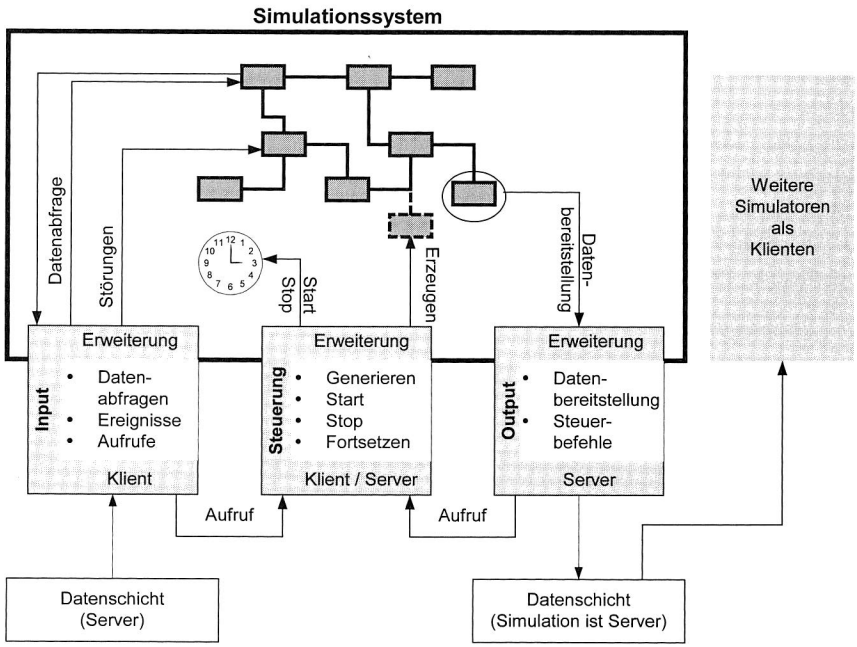


Abb. 35: Aufbau der Darstellungsschicht durch Erweiterung eines Simulationssystems mit den Blöcken Input, Steuerung und Output

Zusätzlich werden die Simulationswerkzeuge auch als Dienstleister (Server) angesehen, die ihre Dienste innerhalb der Architektur anbieten. Demzufolge muss zwischen Datenströmen unterschieden werden, die in die Darstellungsschicht gelangen (Input)

und solchen, die als Ergebnisse anzusehen sind (Output). Konzeptionell ist auf der Inputseite das Repräsentantenkonzept vorzusehen, wogegen die Outputseite als Variante der Datenschicht umzusetzen ist, da die gewonnenen Ergebnisse auch anderen Anwendungen zur Verfügung stehen sollen. Dies entspricht dem Äquivalenzkonzept der Datenklassen P, D und E aus Kap. 3.3.

Um die Klienten-Funktionalität der Simulation zu ermöglichen, ist es notwendig, dass das eingesetzte Simulationswerkzeug über Schnittstellen verfügt, die es ihm erlauben als Klient innerhalb der Architektur zu agieren. Dies kann entweder über die direkte Anbindung an eine Programmiersprache erfolgen oder über eine Schnittstelle (RPC, *Shared Memory* u.a.) des Simulators, die die Kommunikation mit einem externen Prozess ermöglicht. Durch ihn können die vom Simulator empfangenen Informationen in entsprechende Architektur-Aufrufe umgesetzt werden. Außerdem muss es möglich sein, Daten beim Eintritt eines Ereignisses asynchron in das Simulationswerkzeug zu übertragen. Dies ist nötig, um Informationen über Störungen der Produktion (Ereignismeldungen) an den Simulator weiterzuleiten.

Die Rolle der Simulation als Dienstleister (Server) erfordert ferner eine aktive Steuerung des Verhaltens des Simulators. Das eingesetzte Simulationswerkzeug muss es über seine Schnittstellen erlauben, ein Modell zu generieren, dieses zu starten, den Simulationslauf zu steuern und Informationen gezielt abzugreifen. Es sind entsprechende Erweiterungen am Simulationssystem vorzunehmen sowie ggf. entsprechende Bausteine zu realisieren, die die Architektur ausschließlich aus Modellierungs- und Anwendungssicht nutzbar machen. Dieser Schritt muss unter Berücksichtigung der Erweiterungsfähigkeiten marktgängiger Simulationswerkzeuge durchgeführt werden (Abb. 35).

4.6 Kopplung von Simulation mit Systemen der Datenschicht

Die Einteilung der Architektur in drei Schichten hat bisher die internen Funktionsanforderungen der Darstellungsschicht und der Datenschicht beschrieben. Diese sind indirekt über die Anwendungsschicht miteinander verknüpft, was aufgrund der Anwendbarkeit, der Skalierbarkeit und der Implementation wichtiger Funktionen des Gesamtsystems erforderlich ist.

Im Hinblick auf die technische Realisierung des Systems ist weiterhin noch genauer zu klären, wie die Anwendungsschicht zu realisieren ist. Dazu ist zu beschreiben, wie einerseits die Datenquellen, bzw. deren Wrapper, ihre Informationen in das Gesamtsystem einbringen können und andererseits, wie die Simulation solche Informationen wieder aus dem System gewinnen kann. Es sind somit die Schnittstellen zwischen den drei Schichten (Darstellungsschicht - Anwendungsschicht / Datenschicht - An-

wendungsschicht) und deren funktionale Anforderungen innerhalb der Anwendungsschicht festzulegen.

4.6.1 Integration der Systeme der Datenschicht

Um den Klienten der Darstellungsschicht (meist Simulationsmodelle) den transparenten Zugriff auf die Wrapper der Datenschicht (Datenquellen) zu ermöglichen, ist innerhalb der Anwendungsschicht eine Klasse von 'Kommunikationsobjekten' vorzusehen. Ihre Aufgabe ist es, den aus Sicht des Simulationsanwenders geforderten transparenten und robusten Zugriff auf die Datenquellen zu realisieren. Sie bilden das Koppelglied für den Zugriff der Simulation auf die Systeme der Datenschicht.

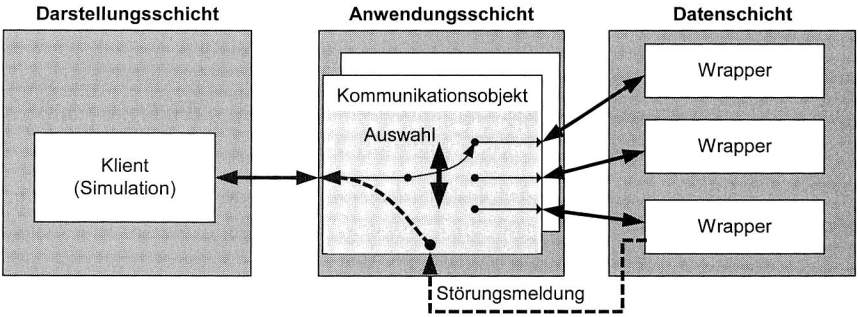


Abb. 36: Aufgaben eines Kommunikationsobjekts

Da die von der Simulation benötigten Informationen auf mehreren Datenschicht-Systemen vorhanden sein können, müssen Kommunikationsobjekte über einen Mechanismus verfügen, der es ihnen ermöglicht, zwischen mehreren unterschiedlichen Datenquellen auszuwählen. Diese Auswahl richtet sich entweder nach den Vorgaben des Klienten/Anwenders oder nach der Verfügbarkeit der vorhandenen Datenquellen.

Um dies zu ermöglichen, werden Kommunikationsobjekte mit mehreren Wrappern verbunden (siehe Abb. 36), aus denen benötigte Daten akquiriert werden können. In der Rückrichtung erfolgen über die Kommunikationsobjekte auch die Ereignismeldungen und die Übermittlung von asynchronen Informationen an die Simulation.

Insgesamt stellen die Kommunikationsobjekte ebenfalls Repräsentanten jeder einzelnen Datenquelle(-ngruppe) innerhalb der Anwendungsschicht der Gesamtarchitektur dar. Um darüber hinaus die Forderung der Ortstransparenz sicherzustellen, den gesamten Datenbestand strukturell und konsistent zu verwalten, muss der Aufbau der Gesamtheit aller Kommunikationsobjekte einem hierarchischen Aufbau folgen. Damit wird es möglich, das Teilsystem 'Anwendungsschicht' aus technischer Sicht als verteiltes System aufzubauen, d.h. es können verschiedene Anwendungs-

schicht-Subsysteme installiert werden, die beispielsweise über verschiedene Anlagenteile, Standorte oder Hierarchieebenen hinweg dezentral die Gesamtfunktion 'Anwendungsschicht' erbringen.

4.6.2 Kopplung der Simulation mit der Anwendungsschicht

In Kap. 4.6.1 wurde die Klasse der Kommunikationsobjekte vorgestellt. Kommunikationsobjekte akquirieren die simulationsrelevanten Daten aus den Wrappern für die Systeme der Darstellungsschicht und speichern diese zwischen. Die Simulation kann an diese Informationen gelangen, indem sie als Klient der Kommunikationsobjekte fungiert. Um hier eine flexible Kopplung zu ermöglichen, wurde für die Kommunikationsobjekte eine Klasse von Stellvertreterobjekten entworfen, die dem Entwurfsmuster *Proxy* entspricht [104]. Diese ermöglichen den simulationsseitigen Zugriff auf die originären Kommunikationsobjekte. Proxyobjekte verhalten sich aus der Sicht der Simulation wie die originären Objekte, haben aber den Vorteil, dass sie lokal zur Simulation einsetzbar sind. Aus diesem Grund sind sie ein wesentlicher Aspekt, was Fragen der Verteilung und Effizienz anbelangt. Darüber hinaus sind sie auch eine funktionale Komponente der Architektur, da sie die Weiterleitung von Ereignismeldungen an die Simulation übernehmen.

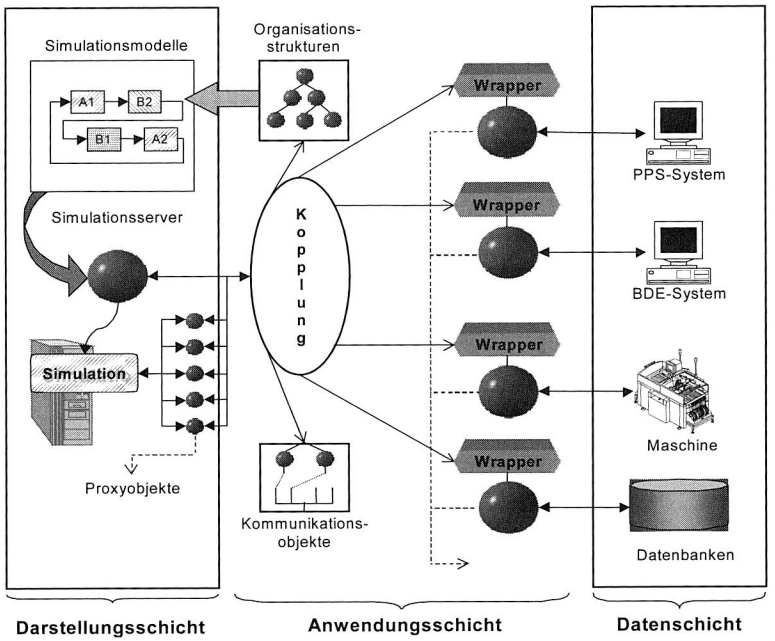


Abb. 37: Kopplung der Simulation mit der Datenschicht

Um die Funktionalität der Simulation innerhalb der Architektur anbieten zu können, wurde eine Klasse konzipiert, die Simulationsdienste kapselt - dies erfolgt aufgrund der Forderung, dass die Simulation ebenfalls als Datenquelle angesehen werden kann. Sie heißt Simulationsserver und entspricht dem Entwurfsmuster Wrapper. Ein Simulationsserver bietet über seine Schnittstellen die grundlegenden Funktionen eines Simulators innerhalb der Architektur an. Dies sind beispielsweise das Starten und das Beenden der Simulation sowie das Transferieren einer Organisationsstruktur aus der Anwendungsschicht in ein Simulationsmodell des gekapselten Simulators. Über den Simulationsserver erfolgt auch die asynchrone Übertragung von Ereignismeldungen in die Simulation.

In Abb. 37 wird veranschaulicht, wie die obengenannten Objekte (Proxyobjekte und Simulationsserver) in Verbindung mit den Objekten der Anwendungsschicht die Kopplung der Simulation mit den Systemen der Datenschicht ermöglichen.

4.7 Repräsentation von Strukturen und Modellkomponenten

Für die Repräsentation der Organisationsstruktur eines Fertigungssystems wird ein hierarchischer Ansatz gewählt. Dies ist naheliegend, da Fertigungssysteme sowie komplexe Simulationsmodelle häufig hierarchisch aufgebaut sind [105,106,107,108]. Das gesamte System wird dabei in eine Reihe von Untersystemen zerlegt, die eine funktionale oder logische Einheit bilden. Das Gesamtsystem kann nun durch das Zusammenwirken der Teilsysteme definiert werden.

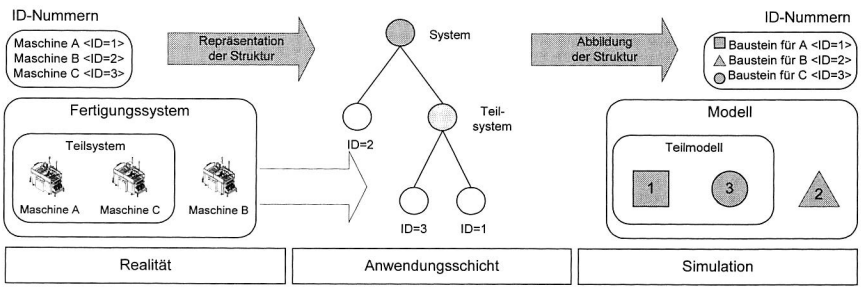


Abb. 38: Repräsentation eines Fertigungssystems und Abbildung auf ein Simulationsmodell

Die oberste Hierarchiestufe eines Fertigungssystems wird durch ein Modellobjekt repräsentiert. Es stellt ein abgeschlossenes System dar und besteht wiederum aus anderen Objekten, durch die Teilsysteme repräsentiert werden. Die Darstellung der Beziehungen (Material- und Informationsfluss) zwischen den Teilsystemen erfolgt ebenfalls durch Objekte. Um aus einem Modellobjekt ein Modell des eingesetzten Si-

mulators zu generieren, müssen auf der Seite der Simulation Komponenten (Bausteine, Objekte, Module u.a.) erstellt werden, die den im Modellobjekt enthaltenen Objekten und Beziehungen entsprechen. Die Generierung des Simulationsmodells erfolgt, indem das Modellobjekt, seine Teilobjekte und die Beziehungen, welche zwischen ihnen bestehen, mit den entsprechenden Komponenten des Simulators abgebildet werden. Für die Darstellung der Teilsysteme eines Modellobjekts werden zwei Arten von Objekten benötigt:

- Objekte, die grundlegende Komponenten des Systems repräsentieren und nicht weiter aufgelöst werden. Diese atomaren Objekte werden Simulationsobjekte genannt und besitzen ein Attribut *ID*, mit dem festgelegt werden kann, welche reale Produktionskomponente zu repräsentieren ist. Dazu ist es nötig, dass jeder Fertigungskomponente eine eindeutige ID-Nummer zugeordnet wird. Die Komponenten des Simulators haben die gleiche ID-Nummer wie die von ihnen modellierte Fertigungskomponente. Bei der Erzeugung eines Simulationsobjekts bekommt dieses die ID-Nummer der Fertigungskomponente, die es repräsentieren soll und hinterlegt diese in seinem ID-Attribut. Bei der Generierung eines Modells aus dem Modellobjekt erfolgt die Abbildung des Simulationsobjekts durch die Komponente des Simulators mit der gleichen ID-Nummer. So werden z.B. Simulationsobjekte mit der ID-Nummer 2 mit der Komponente abgebildet, die ebenfalls die ID-Nummer 2 hat (Abb. 39). Die Vergabe der ID-Nummern erfolgt durch den Anwender.
- Objekte, die komplexe Teilsysteme repräsentieren. Sie werden Netzwerkobjekte genannt und bestehen aus Simulationsobjekten und anderen Netzwerkobjekten. So wird die Bildung von Hierarchien ermöglicht. Durch diese Objekte werden Organisationseinheiten zusammengefasst, die eine logische oder funktionale Einheit bilden, wie z.B. eine Fertigungszelle oder eine Abteilung.

Die obengenannten Objekte reichen aus, um den hierarchischen Aufbau eines Fertigungssystems darzustellen. Da aber zwischen Fertigungskomponenten Verbindungen (Beziehungen) bestehen, über die der Austausch von Informationen oder Material stattfindet, ist es nötig, diese Verbindungen bei der Darstellung einer Organisationsstruktur zu berücksichtigen. Sie können durch gerichtete attributierte Assoziationen zwischen den beteiligten Objekten modelliert werden [109]. So kann eindeutig festgelegt werden, zwischen welchen Komponenten der Fluss stattfindet und welches Gut dabei ausgetauscht wird. Die Identifizierung des transportierten Gutes (Information/Material) erfolgt, indem jedem möglichen Fluss eine Typen-Nummer zugeordnet wird. Diese werden vor der Repräsentation des Fertigungssystems vergeben und dienen der Abbildung des Material- oder Informationsflusses auf die entsprechende Material- oder Informationsfluss darstellende Komponente des Simulations-

werkzeugs. Diese müssen deshalb ebenfalls über eine Typen-Nummer verfügen. So wird z.B. ein Fluss mit der Typen-Nummer 1 durch die Simulator-Komponente mit der gleichen Typen-Nummer abgebildet (Abb. 39).

Die Realisierung der obengenannten Assoziationen erfolgt durch Objekte der Klassen Inport, Outport und Link, die ein Attribut namens *Type* besitzen, in dem die Typen-Nummer des repräsentierten Flusses hinterlegt wird. *Inports* repräsentieren Schnittstellen für eingehenden Material- oder Informationsfluss, *Outports* hingegen Schnittstellen für abgehende Flüsse. Die Verbindungen zwischen *Inports* und *Outports* erfolgen durch *Link*-Objekte. Dabei ist zu berücksichtigen, dass Verbindungen nur zwischen Objekten mit der gleichen Typen-Nummer möglich sind, da ein Link nur für den Transport und nicht die Umwandlung einer Information genutzt werden darf. *Inports* und *Outports* sind Bestandteile der Teilsysteme eines Modellobjekts (Netzwerk- und Simulationsobjekte). Vom Modellobjekt selber werden sie nicht benötigt, da dieses ein abgeschlossenes System repräsentiert.

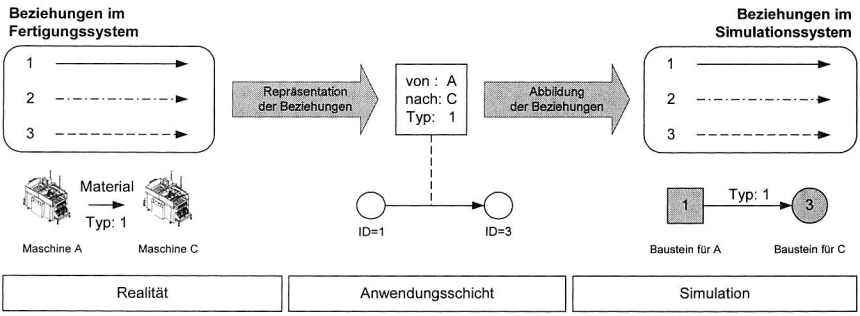


Abb. 39: Repräsentation und Abbildung der Beziehungen

Um über die dargestellten Organisationsstrukturen eine Anbindung an die realen Fertigungskomponenten zu ermöglichen, müssen Simulations-, Netzwerk- und Modellobjekte die Funktionalität von Kommunikationsobjekten beinhalten.

4.8 Schnittstelle zwischen Datenbeständen und Simulator

Es wurde bereits beschrieben, dass die zu verwaltenden Informationen umfangreich sind. Analog zu dem Problem eines Simulationsexperten, sich beim Entwurf eines konzeptionellen Modells zunächst einen Überblick zu verschaffen und anschließend die 'relevanten' Systemelemente auszuwählen, ist dieses Problem auch bei den durch dieses System bereitgestellten Informationen gegeben. Im Gegensatz zu realen oder geplanten Systemen liegen die Informationen jedoch wohl strukturiert und als simulationsbezogene (Primär-)Daten vor. Hierbei sind die Schlüsselfunktionen ei-

nerseits in der Organisation und andererseits in einer Navigationskomponente zu sehen.

4.8.1 Komponente: Navigator

Navigation bezeichnet im folgenden *einen Prozess zur Nutzbarmachung von Informationen der Verwaltungsschicht für Datenbeschaffungszwecke während der Modellierung oder Nutzung eines Simulationsmodells.* Dieses stellt bei vorgehend beschriebener Strukturierung der Informationsbestände keinen nennenswerten Zusatzaufwand dar, weil die bereitgestellten Informationen ohnehin explizit aufbereitet werden müssen, z.B. im Rahmen der Registrierung eines Wrappers bei der Verwaltungsschicht.

Bei der Navigation durchläuft der Anwender, beginnend von der Wurzel eines Hierarchiebaumes, die Informationsstruktur ebenenweise - bei der Sichtweise "Struktur" wäre dies z.B. Unternehmen → Werk1 → Halle2 → Bereich3 → Zelle4 → Maschine5 usw. Dieser Prozess endet an einem Blatt des Baumes oder wenn ein Knoten ausgewählt wurde. Im zweiten Fall gelten alle Elemente ab diesem Knoten als selektiert, womit umfangreichere Funktionsblöcke in einem Schritt verwendet werden können. Am vorherigen Beispiel würde die Auswahl von Zelle4 (etwa ein Drehzentrum) somit alle zu dieser Zelle gehörenden Maschinen und Komponenten (Drehmaschine, Puffer, Übergabe zur Materialbereitstellung → FTS usw.) auswählen.

Die mögliche Registrierung einer Datenquelle in mehreren Bäumen soll neben der Strukturierung besonders bei der Navigation einen Sprung zwischen den Hierarchien ermöglichen. Damit kann beispielsweise in der Hierarchie "Produkt" begonnen werden. Wird eine problemrelevante Detaillierungsebene erreicht, kann dann auf die Hierarchie "Ressource" gewechselt werden. Die hierzu notwendigen Navigationsmarken können jederzeit aus dem Hierarchiebaum "Prozess" oder aus einem indizierten Datenquellenverzeichnis ermittelt werden, in dem die entsprechenden Datenquellenreferenzen durch effiziente Suchalgorithmen gefunden werden können. Insgesamt muss die Navigationskomponente die technische Realisierung der in Kap. 4.2.2 aufgestellten Forderungen, etwa das Verwalten der Navigationspfade und das Vermeiden von Zyklen, erfüllen.

Weiterhin muss die Navigationskomponente eine Ablage ("*scratch-board*") verwalten können, in der alle ausgewählten Datenquellen einschließlich der notwendigen Verwaltungsinformationen zwischengespeichert werden können. Dieses scratch-board soll ebenfalls einen hierarchisierten Überblick über alle ausgewählten Komponenten geben. Im Falle der Modellbildung spiegelt diese Hierarchie somit den Grundzug des

zu erstellenden Simulationsmodells wider und enthält demzufolge Entsprechungen der Datenklasse S.

Innerhalb des scratch-boards kann der Anwender nun Teilbäume verschieben, löschen oder erweitern, sodass innerhalb der betrachteten Systemgrenzen eine weitgehend vollständige Blockstrukturbeschreibung mit variablem Detaillierungsgrad des Simulationsmodells erfolgt (für diese Vorgehensweise wurde bereits der Begriff der datengetriebenen Modellierung eingeführt).

4.8.2 Komponente: Modellgenerator

Im Anschluss an die Navigation muss das Navigationsmodul zur Modellbildung eine Verbindung zum Wrapper des jeweils verwendeten Simulationssystems aufbauen. Durch ein ebenfalls zu realisierendes Generatormodul innerhalb des Navigators wird der Hierarchiebaum des scratch-boards ebenenweise durchlaufen und ein Simulationsmodell mit den entsprechenden Komponenten (Objekte oder Netzwerke) aufgebaut.

Das gewählte Vorgehen ist eine Erweiterung des Ansatzes von [110], der die Modellbildung über Steuerdateien (Skripte) vorsieht. Der dort präsentierte Weg über Dateien wäre jedoch hinsichtlich einer offenen und verteilten Architektur kaum geeignet, da solche "Schnittstellen" meist unidirektional sind und demzufolge beispielsweise der asynchrone Nachrichtenaustausch hierüber nicht stattfinden kann.

Stattdessen wird der Modellgenerator an einen simulationssystemseitigen Wrapper gekoppelt. Dieser Wrapper folgt dem bereits beschriebenen Konzept der Wrapper von Datenquellen und soll zusätzlich den formulierten Anforderungen nach Steuerbarkeit und (a)synchronem Ereignisaustausch genügen, d.h. im hier beschriebenen Anwendungsfall das extern gesteuerte Erzeugen von Simulationsmodellen erlauben.

4.8.3 Komponente: Funktionen und Bausteine im Simulator

Die Bausteine innerhalb des Simulationssystems nehmen hinsichtlich der Nutzung der gesamten Architektur aus Anwendersicht eine zentrale Rolle ein. Ihre Aufgabe ist es, die in der Anwendungsschicht verwalteten Repräsentanten der Datenschicht als Simulationskomponenten darzustellen. Das gesamte System wird somit aus dem Simulator heraus nutzbar.

Die Bausteine sind abhängig vom genutzten Simulationssystem geeignet auszulegen: Werden baustein- und objektorientierte Modellierungsparadigmen verwendet (z.B. eM-Plant, Taylor, Quest), sollen die bestehenden Bausteine oder Klassen erweitert werden. Dadurch wird eine konsistente Begrifflichkeit gewährleistet, weil der

Anwender die bestehende Funktionalität weiterverwenden und bedarfsweise die neuen Architekturfunktionen nutzen kann.

Arbeitet das Simulationssystem auf der Basis einer prozeduralen Beschreibung (z.B. AutoMod, Simplex III, GPSS), sind entsprechende Zusatzfunktionen i.S. einer API (*application programming interface*) anzubieten.

Die Integration der Kommunikationsfunktionalitäten ist unter Beachtung der einheitlichen Verwendbarkeit systemspezifisch vorzunehmen.

4.9 Zusammenfassung

Die vorhandenen Informationsstrukturen stimmen meist nicht mit den bestehenden Organisationsstrukturen überein, was sich immer mehr als Hemmnis für den erfolgreichen Einsatz der Simulation herausstellt: Einerseits sind die zur Simulation notwendigen Basisdaten nicht in einer einheitlichen Form verfügbar und andererseits nicht entsprechend organisiert, um hieraus die Grundzüge der Simulationsmodellstrukturen ableiten zu können.

Hieraus ergibt sich als Aufgabe und Ziel dieser Arbeit die Entwicklung einer Architektur zur flexiblen Kopplung eines marktgängigen Simulationswerkzeugs mit Datenquellen zum Zweck des Aufbaus, der Initialisierung und Parametrierung von Simulationsmodellen während der Planung und dem Betrieb von hochautomatisierten Produktionssystemen.

Hierzu ist das Simulationssystem so zu erweitern, dass die Nutzung der Architektur für den Benutzer keinen wesentlichen Programmier– bzw. Konfigurationsaufwand bedeutet. Neben der Nutzung dedizierter Datenbanken, was dem heutigen Stand der Technik entspricht, liegt der Schwerpunkt auf der Möglichkeit, einzelne Modellkomponenten des Simulationsmodells direkt durch korrespondierende reale Anlagenobjekte zu parametrieren. Die dazu notwendige kommunikationstechnische Integration soll durch ein flexibel konfigurierbares Kopplungsmodul, welches als Mittler zwischen Simulationssystem und unterschiedlichen Datenquellen fungiert, erfolgen.

Dadurch ist es möglich, der Simulation einen weitgehend transparenten Zugriff auf die typischerweise verteilten Datenbestände eines Unternehmens, unabhängig vom jeweiligen Standort und den internen Kommunikationsprotokollen der Datenobjekte, zu ermöglichen.

Die zu erwartenden Nutzungspotentiale dieses Ansatzes liegen somit im Bereich der Modellerstellung und der Datenakquisition für eine prozessgekoppelte (online) Simulation, zu dessen Zweck die akquirierten Daten durch flexibel einsetzbare Über-

wachungskomponenten betriebsbegleitend innerhalb des Simulationsmodells oder innerhalb eines externen Moduls ausgewertet werden können. Hierdurch werden die Einsatzpotentiale und der Nutzen des Gesamtsystems überaus deutlich.

5 Bewertung ausgewählter Middlewarearchitekturen aus Sicht der Simulation

Die derzeit in den Unternehmen vorzufindenden Informationssysteme stellen im wesentlichen erst eine Annäherung an die Vision dar, Informationen einheitlich und unternehmensweit bereitzustellen.

Der Grund hierfür liegt ursächlich in der Vielfalt der verwendeten Rechner-, Betriebs- und Netzwerksysteme, die eine umfassende Integration kaum zulassen. Die zur Entschärfung dieser Situation notwendigen Vorgehensmodelle sind entweder nicht vorhanden oder stellen bestenfalls hochspezialisierte Individuallösungen dar. Demzufolge sind Anwendungen meist technisch und räumlich begrenzte Werkzeuge, die aufgrund von Systemgrenzen außerhalb ihres ursprünglichen Nutzungsumfelds nicht eingesetzt werden können. Diese Einschränkung gilt nicht nur zwischen Unternehmen, sondern meist schon zwischen den Anwendungen einzelner Abteilungen, obwohl die zu erwartenden Potentiale einer Integration sowohl organisatorisch als auch technisch vielversprechend sind. Als Kernaussage ergibt sich jedoch allgemein, dass Organisationsstrukturen und Informationsstrukturen nicht kompatibel sind. Eines der in diesem Zusammenhang am meisten diskutierten Begriffe ist dabei sicherlich das "Business Process Reengineering", bei dem bestehende Geschäftsprozesse umfassend überarbeitet und in rechnergestützte Systeme integriert werden sollen [111]. Die genannten organisatorischen Potentiale sind dabei sehr gut am Beispiel der Integration von PPS-Systemen in Unternehmen zu erkennen. Erfahrungsberichte zeigen hier, dass die organisatorischen Vorbereitungen am PPS-Projekt einen höheren Zeitanteil erfordern, als die eigentlich Installation [112]. Aufwandstreiber sind üblicherweise alte, redundante und inkompatible/proprietäre Datensysteme und -bestände sowie veraltete Hardware. Diese Heterogenitäten müssen jeweils unter großem Aufwand bereinigt und aktualisiert werden, bevor die verbesserte Funktionalität der Neuinvestition effektiv genutzt werden kann.

5.1 Middleware

Auf der technischen Seite wird der Heterogenität von Hard- und Software und der zunehmenden Verteiltheit von Datenbeständen seit der zweiten Hälfte der 1990er Jahre durch das Konzept der *Middleware* begegnet. Dieses sieht vor, dass zwischen den spezifischen Anwendungen und genutzten Betriebsystemen eine zusätzliche Softwareschicht eingeführt wird. Diese hat die Aufgabe, die spezifischen Eigenschaften des Betriebssystems aus Sicht des Anwenders und Anwendungsentwicklers zu verbergen.

Middleware stellt für den Benutzer eine transparente Software dar, die zwischen den Anwendungen und dem Betriebssystem agiert. Dementsprechend hat eine Middleware die Aufgabe, die Anwendungen von den benutzten Schnittstellen des Betriebssystems und der verwendeten Hardware zu isolieren und auf Basis von Remote Procedure Calls (RPCs) die Interaktion der auf diesem System implementierten Anwendungen zu ermöglichen.

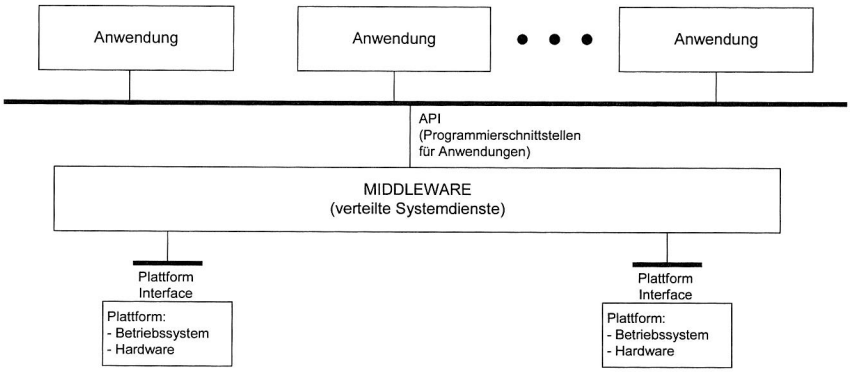


Abb. 40: Middleware als Dienst zur Vermittlung zwischen Anwendungen und Plattformen

Dem Entwickler bietet sich damit die Chance, losgelöst von der Komplexität der informationstechnischen Gegebenheiten (Betriebssystem, Netzwerk, Formate usw.) sich auf die Lösung der anwendungsbezogenen Probleme zu konzentrieren und damit einen messbaren Mehrwert zu erreichen. Für den Anwender bedeutet dies eine Investition in die Anwendungssoftware, ohne bei einer möglicherweise erforderlichen Folgeinvestition an die darunter liegenden Computersysteme (Hardware) gebunden zu sein. Ebenso ist auf diese Weise sichergestellt, dass ähnliche Anwendungsprogramme von u.U. unterschiedlichen Herstellern gemeinsam eine erweiterte Funktion erbringen können.

Aus diesen Gründen wird Middleware für neue Anwendungen zumeist wichtiger sein als Betriebs- und Netzwerksysteme, weil allgemein davon ausgegangen wird, dass Middleware die bisher genutzten (nicht verteilten) Funktionen eines Betriebsystems aus Anwendersicht ersetzt und die verteilte Funktionalität des Netzwerks produktiv erschließt [113]. Hier wird auch von einem "digitalen Nervensystem" für ein Unternehmen gesprochen [114]. Durch die Entwicklung von Standards wie CORBA, DCOM, EntireX, InfoBus usw. stehen dazu offene, leistungsfähige und homogene Plattformen zur Verfügung. Die aktuellen Entwicklungen auf dem Markt der Middlewares zeigen deutlich, dass Middleware mehr ist als ein Adapter zur transparenten Verbindung verteilter Anwendungen. Vielmehr werden durch eine Middleware unter-

nehmensweite Applikationen und damit auch Organisationen verbunden. Entsprechend wird bis 2002 ein Markt von etwa 7 Mrd. US$ erwartet [115]. Auch die rege Beteiligung von Unternehmen an Organisationen, wie z.B. X/Open, OSF und OMG belegen den hohen Stellenwert, den die Industrie dem Middlewaregedanken beimisst.

5.2 Frameworks

Eine standardisierte Middleware bietet für Anwendungsentwickler bereits eine Konzentration auf die zu realisierenden Aufgaben. Darüber hinaus wird in diesem Zusammenhang das Konzept der Frameworks diskutiert. Frameworks stellen eine Softwareumgebung bereit, die die Anwendungsentwicklung weiter vereinfacht. Sie umfassen mindestens eine Middleware sowie eine Anzahl von Werkzeugen, die von Anwendungen genutzt werden können. Die Werkzeuge ("Tools") stellen dabei universelle Funktionen zur Verfügung, die die Verwendungen des gesamten Frameworks vereinfachen. Die Integration von Tools in ein Framework erfolgt dabei durch Kontrollflüsse und/oder Daten. Der Austausch erfolgt durch die genutzte Middleware. Eine neue Anwendung ergibt sich somit aus der Integration verschiedener Tools, die über eine Middleware zu einem Anwendungssystem zusammengestellt werden.

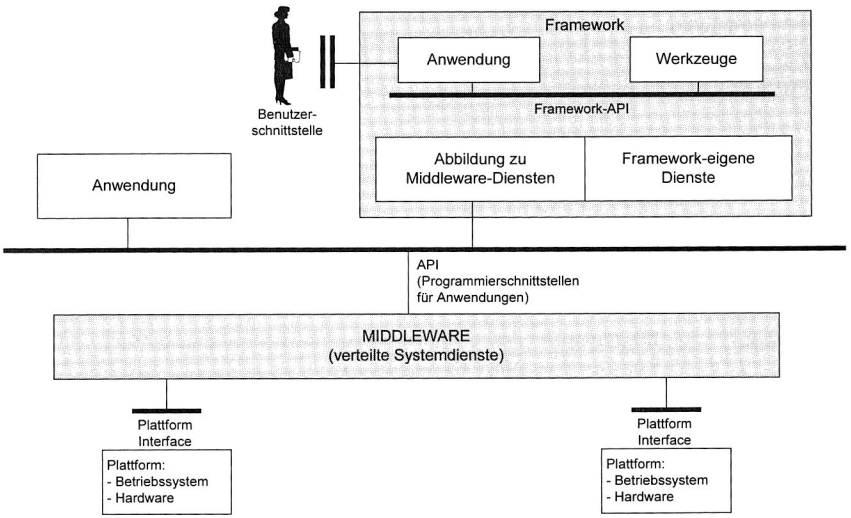


Abb. 41: Grundlegende Architektur eines Frameworks

Das derzeit wohl bekannteste Framework ist die MS-Office Produktfamilie. Neben Tools wie einer Textverarbeitung stehen eine Tabellenkalkulation, eine Datenbank, ein Präsentationsprogramm und ein Projektplaner zur Verfügung. Diese Anwendun-

gen gewinnen durch eine zusätzliche Programmiersprache (Visual Basic) und eine MS-eigene Middleware (D)COM den Status von Tools und lassen damit den Aufbau von Frameworks zu. In [116] wird beispielsweise beschrieben, wie unter ausschließlicher Verwendung der genannten Anwendungen ein PPS-System erstellt worden ist. Dies zeigt die konzeptionelle Leistungsfähigkeit der beschriebenen Middleware- und Frameworkkonzepte auf.

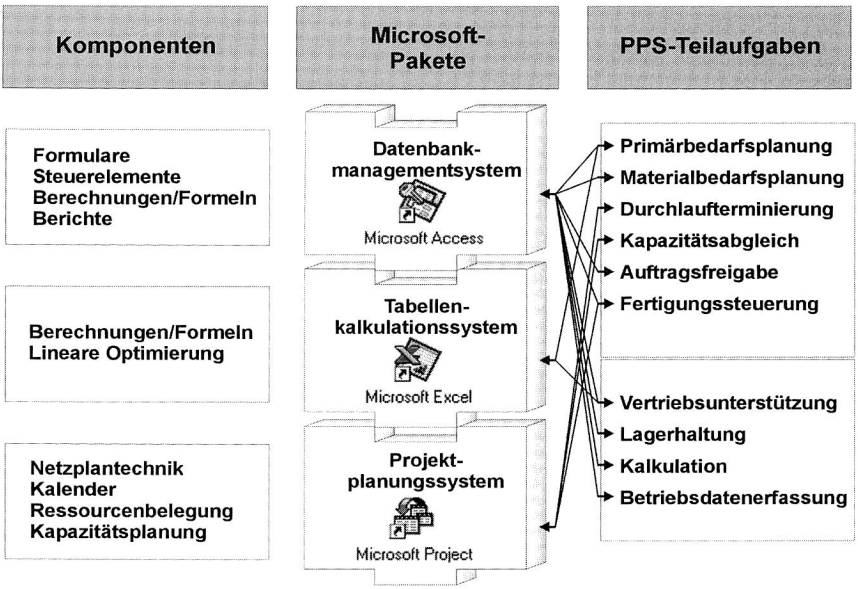


Abb. 42: Beispiel eines Frameworks zur Realisierung eines PPS-Systems unter Verwendung von MS-Office Komponenten (nach [116])

5.3 Diskussion verschiedener Middlewarearchitekturen

Die Aufgabe einer Middleware, eine Plattformunabhängigkeit anzubieten, verspricht nicht nur in der Nutzung erhebliche Potentiale. Auch Softwarehersteller haben erkannt, dass für Middlewares selbst ein großes Marktpotential besteht. Folglich stehen mehrere Middlewares in Konkurrenz zueinander. Es ist daher notwendig, die jeweiligen Grundkonzepte unter dem besonderen Blickwinkel der in Kap. 4 aufgestellten Anforderungen zu diskutieren und auszuwählen. Die Diskussion konzentriert sich auf die Architekturen, welche hinsichtlich Preis, Leistung, Verbreitung und Nutzbarkeit derzeitig die höchste Marktdurchdringung aufweisen (DCOM und CORBA), bzw. die aufgrund ihres primären Einsatzzweckes berücksichtigt werden müssen (HLA).

5.3.1 DCOM

Das *Component Object Model* (COM) wurde 1993 von Microsoft eingeführt und ist seitdem in diesem Unternehmen die Schlüsseltechnologie zur Entwicklung komponentenbasierter Software.

Bei COM handelt es sich um Microsofts Variante eines Modells, dass die Zusammenarbeit verschiedener Applikationen ermöglichen soll. Es handelt sich um ein objektbasiertes Programmiermodell nach dem Client/Server-Prinzip. Hauptaufgabe des COM ist es, einem Client zu ermöglichen, auf die Dienste eines Servers zuzugreifen, unabhängig davon wer sie wann und mit welcher Sprache erstellt hat (sowohl die Clients als auch die Server). Um dies zu erreichen wird innerhalb des COM ein binäres Format festgelegt, wie die auszutauschenden Objekte im Speicher abzulegen sind. Dieses festgelegte Format ermöglicht es jeder Programmiersprache, die in der Lage ist, Objekte in der geforderten Form im Speicher abzulegen, COM-Objekte zu erstellen. Ein COM-Objekt stellt seine Dienste einem anderen COM-Objekt durch ein (oder auch mehrere) Interface(-s) zur Verfügung. Die Module in denen COM-Objekte enthalten sind werden als COM-Server bezeichnet. Der Programmierer sieht daher nur Objekte mit ihren Schnittstellen (Abb. 43).

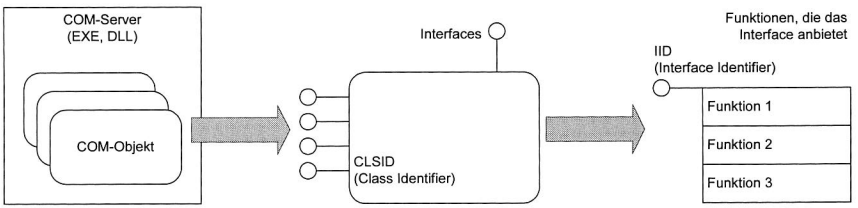


Abb. 43: Das COM-Programmiermodell (nach [137])

Die Interfaces bilden die Schnittstelle zu einem COM-Objekt, da nur durch sie Aktionen mit einem solchen Objekt durchgeführt werden können. Bei diesen Interfaces handelt es sich um Gruppen inhaltlich gleicher Funktionen.

Durch definierte Aufrufmechanismen (in-process, local oder remote server) wird ein Zugriff auf die Funktionen des Interfaces eines COM-Objektes ausgeführt. Wie oben erwähnt beinhaltet das Interface Funktionen, die zum Zugriff auf ein COM-Objekt benötigt werden. Rein softwaretechnisch handelt es sich hierbei um einen Zeiger auf eine Tabelle, deren Inhalt aus Zeigern auf die Funktionen der aktuellen Implementierung der Interfacefunktionen des COM-Objektes besteht.

Kurz nach Einführung von COM wurde diese Architektur um die Fähigkeit von verteilten Objekten erweitert: *Distributed COM* (DCOM - Abb. 44) erlaubt es, Komponenten auf verschiedene Rechner im Netzwerk zu verteilen [117]. Wiederverwend-

barkeit, Sicherheit (Zugriffsschutz und Zugriffsrichtlinien), Verfügbarkeit (Redundanz, Ausfallsicherheit), Skalierbarkeit (Performance, Lastverteilung) und Wartbarkeit (einschließlich Versionsverwaltung) sind dabei wichtige neue Eigenschaften, die von DCOM aufgegriffen werden.

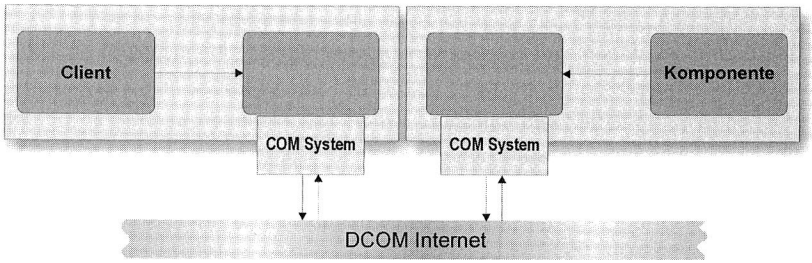


Abb. 44: Aufbau verteilter Objekte durch Distributed COM (DCOM)

Durch die hohe Verbreitung von Microsoftprodukten kann DCOM de-facto als Industriestandard angesehen werden. Anwendungen reichen daher von komplexen Geschäftsregelkomponenten, z.B. eigene Client-Programme und JavaScripte bis zu Web-Seiten und ActiveServer Pages bei Web-Servern. Entsprechend existiert auch eine Vielzahl an Entwicklungswerkzeugen und eine breites Entwickler-Know-how. Ebenso findet DCOM mittlerweile punktuell auch Verbreitung auf anderen Plattformen, speziell auf den bekannteren Unix-Derivaten wie Solaris und Linux.

Somit stellt (D)COM nach der vorgehend genannten Beschreibung eine Middleware dar, die aufgrund ihres Ursprungs derzeit die größte Marktdurchdringung aufweist und den erprobten Stand der Technik widerspiegelt.

5.3.2 CORBA

1989 wurde die Object Management Group (OMG) als Konsortium von 11 namhaften Firmen gegründet und wuchs bis Ende 1999 auf über 900 Firmen an. Dabei handelt es sich um eine Vielzahl bekannter Unternehmen und Einrichtungen aus den Bereichen Informationstechnologie, Forschung und Finanzen. Dieser Verbund, dessen Mitglieder in die Kategorien Hardwarehersteller, Softwareentwickler und Endbenutzer unterteilt werden, hat sich durch das Erarbeiten von Standards zum Ziel gesetzt, die Entwicklung von Technologien für den Einsatz in objektorientierten verteilten Systemen in folgenden Punkten zu beeinflussen [118]:

- Vereinheitlichung des Vokabulars für die Beschreibung verteilter Systeme und Definition der dadurch beschriebenen Sachverhalte

- Vereinheitlichungen von Programmierschnittstellen, so dass die Zugriffe in einem auf OMG-Standards basierenden System, unabhängig vom eingesetzten Betriebssystem, immer auf die gleiche Art notiert werden können
- Entwurf einer universellen Kommunikationsplattform, durch die die *Interoperabilität* verschiedener, unabhängig voneinander entwickelter Systeme möglich ist
- Definition von Werkzeugen für die Verteilung von Softwarekomponenten innerhalb eines Netzwerks und deren anschließende Verwaltung
- Definition von wiederverwendbaren Vorgehensweisen, die bei der Gestaltung einer Vielzahl von Anwendungen eingesetzt werden können

OMG-Standards verstehen sich als Richtlinien, die aussagen, welche Eigenschaften und welches Verhalten ein konformes Produkt haben muss. Dieser OMG-Standard dient als übergeordnete Beschreibung einer Architektur (Object-Management Architecture OMA), dessen Kern, der Object Request Broker (ORB), die Spezifikationsgrundlage von CORBA (der Common Object Request Broker Architecture) ist.

Die Object Management Architecture (OMA)

Grundlage aller Standardisierungsaktivitäten der OMG ist die *Object Management Architecture* (OMA) [119]. In ihr werden Objekte als die grundlegenden Bausteine einer verteilten Anwendung festgelegt. Innerhalb der OMA wird zunächst ein allgemeines Objektmodell definiert, in dem die Grundbegriffe für den Standard definiert werden. Die OMA unterteilt die Bestandteile einer verteilten Anwendung in Komponenten. Konkret handelt es sich jeweils um eine Menge von Objekten, die gemeinsam eine bestimmte Aufgabe erfüllen (z.B. Kommunikation von Objekten untereinander, Auffinden von Objekten im Netz usw.). In separaten Standards werden von der OMG eine Reihe von Komponenten der OMA identifiziert, indem die Funktionalität und die Schnittstellen der beteiligten Objekte spezifiziert werden.

Diese Komponenten werden von der OMA einer der folgenden fünf Schichten zugeordnet (siehe Abb. 45): *ORB, Object Services, Common Facilities, Domain Services, Application Objects* [119].

Der Object Request Broker (ORB)

Der ORB übernimmt in der OMA-Referenzarchitektur die Rolle des Kommunikationsvermittlers. Das ist diejenige Komponente, die von Objekten zur Kommunikation mit anderen Objekten desselben oder auch eines unterschiedlichen Adressraumes benutzt wird. Der ORB leitet Operationsaufrufe – für das aufrufende Objekt transparent – an einen Zielrechner und dann an das betreffende Objekt. Dies geschieht unabhängig davon, welches Betriebssystem dort verwendet wird und in welchen Pro-

grammiersprachen die beteiligten Objekte implementiert wurden. Diese Eigenschaft wird als Plattforminvarianz (unabhängig von Betriebsystem und der Hardware) und Ortstransparenz (der Objektzugriff erfolgt unabhängig von der Kenntnis, wo sich das Objekt befindet) bezeichnet.

Object Services

Unter dem Begriff *Object Services* werden diejenigen Komponenten der OMA zusammengefasst, die elementare, betriebssystemähnliche Funktionen innerhalb der Architektur anbieten. Sie realisieren grundlegende Dienste (Lokalisierung eines Objekts, Ereignisbehandlung u.a.), die den Gebrauch und die Implementierung von Objekten erleichtern, sind aber nicht auf ein fachliches Anwendungsgebiet spezialisiert. Die OMG definiert nur die Funktionalität der *Object Services*. Die konkrete Form ihrer Realisierung wird dem Hersteller überlassen. Diese Services bilden die Kernfunktionalität, die den ORB als Middleware präsentiert.

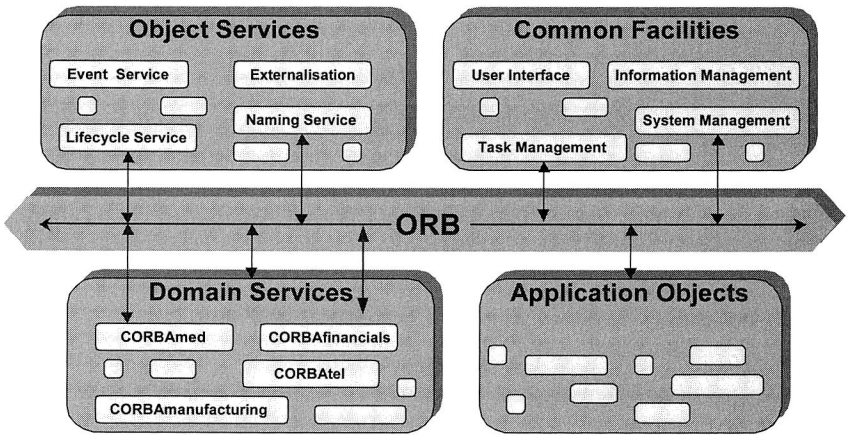


Abb. 45: Object Management Architecture [118]

Common Facilities

Common Facilities sind komplexe funktionale Einheiten, die für verschiedene Anwendungen bzw. Anwendungsgruppen nützlich sind. Sie verfügen über Konfigurationsmechanismen, mit denen sie an unterschiedliche Aufgabenstellungen angepaßt werden. Die *Common Facilities* bauen auf den *Object Services* auf, sind aber stärker spezialisiert als diese. Beispiele für *Common Facilities* sind graphische Benutzeroberflächen (*Graphical User Interface*), Dienste zur Druckersteuerung, zur Dokumentenverwaltung, Datenbanken und E-Mail [118].

Die Verfügbarkeit von Common Facilities, bzw. die einfache Möglichkeit, diese zu nutzen, erschliesst CORBA als Middlewareplattform für Frameworks. Common Facilities werden häufig auch als Horizontal Facilities bezeichnet.

Domain Services

Diese auch als *Vertical Market Facilities* bezeichneten Konzepte sind Lösungen für spezielle Anwendungsgebiete, die im Bezug auf ihren prinzipiellen Aufbau durch Standards der OMG abgedeckt sind. Beispiele für *Domain Services* sind Frameworks für das Finanzwesen (*CORBAfinancials*), für das Gesundheitswesen (*CORBAmed*), für die Fertigungstechnik (*CORBAmanufacturing*) usw. *Domain Services* werden durch die Kombination von *Object Services* und *Common Facilities* definiert [118].

Application Objects

Unter dieser Bezeichnung finden sich in der OMA alle Objekte, die die eigentliche anwendungsspezifische Funktionalität erfüllen. Ein *Application Object* ist in der OMA demzufolge das, was vereinfacht als Anwendung bezeichnet wird. Für die Erfüllung ihrer fachlichen Aufgaben nehmen *Application Objects* die Hilfe von *Object Sevices* und *Common Facilities* in Anspruch. Sie stellen eine individuelle Lösung für ein spezielles Problem dar und sind daher kein Bestandteil der Standardisierungsprozesse der OMG.

5.4 HLA - ein Framework für verteilte Simulation

Geringere Ressourcen bei gleichzeitig steigender Unsicherheit der Entscheidungsträger sind seit Beginn der 1990er Jahre nicht nur eine veränderte Randbedingung in der Industrie. Auch die Entwicklung und der Betrieb militärischer Simulationsanwendungen unterlag seitdem drastischen Einsparungsmaßnahmen. Dieser Herausforderung begegnete das amerikanische Verteidigungsministerium (DoD - Department of Defense) durch die Initiative, die vielfältigen Simulationsanwendungen und -kompetenzen kosteneffektiv zu konzentrieren. Es wurde erkannt, dass die Grundlage hierfür eine Informationsarchitektur sein muss, die Weiterverwendbarkeit der verwendeten Anwendungen genauso unterstützt wie deren verteilten Einsatz.

Das Ergebnis war der Entwurf der High-Level-Architecture (HLA) durch das DMSO (Defense Modeling and Simulation Office). Diese erlaubt es, einzelne Simulationen in einen umfassenderen Simulationskontext i.S. einer verteilten Simulation zu integrieren (Abb. 46). Analog zu CORBA ist HLA nicht als Implementation einer Infrastruktur, sondern vielmehr als Softwarearchitektur aufzufassen. Daher umfasst HLA auch eine Vielzahl unterschiedlicher Implementierungen, deren gemeinsame Grundlage dokumentiert und teilweise bereits standardisiert ist. Der HLA-Standard - wie vom DoD

vorgeschlagen - besteht aus drei Teilen: HLA-Regeln, einer Meta-Sprache zur Beschreibung des Aufbaues komplexer, verteilter Simulationsmodelle (dem Object Model Template - OMT) sowie einer Interfacespezifikation, die das Zusammenspiel zwischen einzelnen Komponenten eines "Gesamtmodells" und der darunter liegenden Organisationsschicht, der *run-time-infrastructure* (RTI), festlegt.

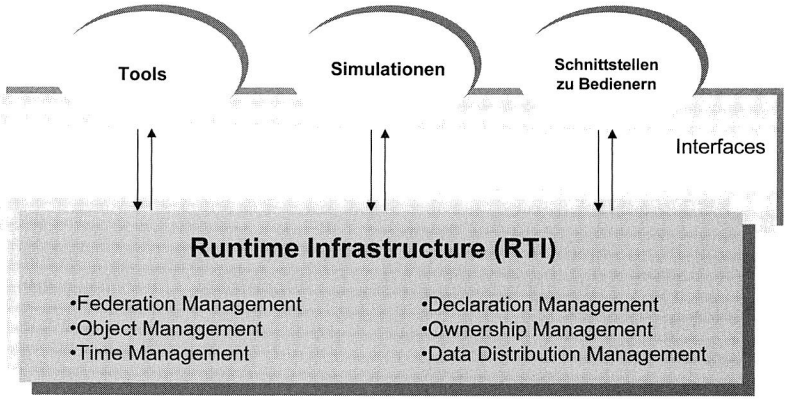


Abb. 46: Funktionale Sicht auf die High-Level-Architectur (HLA)

5.4.1 HLA-Regeln

Die HLA-Regeln legen die Prinzipien und Konventionen fest, denen die einzelnen Komponenten (*federates*) folgen müssen, um gemeinsam in einer *federation* die erwartete Funktion zu erbringen. Hierbei wird beschrieben, wie die Interaktionen zwischen den *federates* während der Ausführung einer *federation* zu organisieren sind, und welches organisatorische Verhalten bei der Implementierung durch die Simulationsexperten berücksichtigt werden muss.

Verlangt wird, dass die Simulationsfunktionalität jedes einzelnen *federates* gemäß der OMT-Definition dokumentiert in einem *simulation-object-model* (SOM) hinterlegt wird. Die Mindestanforderung dazu umfasst zwingend alle Operanden des Simulationsobjekts: Zur Laufzeit muss das Simulationsobjekt in der Lage sein, die hiermit verbundenen Attribute zeit- und zustandsrichtig zu verwalten, zu aktualisieren und/oder bedarfsweise auch die Eigentümerschaft (Zuständigkeit) eines Attributs zu übernehmen oder abzugeben.

Weiterhin verlangen die Regeln, dass analog zum SOM eines federates auch eine *federation* in einem *federation-object-model* (FOM) dokumentiert wird. Hier wird beschrieben, welche *federates* durch welche Informationen und Interaktionen semantisch konsistent zusammenarbeiten, also eine federation bilden. Diese Informationen werden während der Planung herangezogen, um im Sinne eines Frameworks die

Einzelkomponenten geeignet zusammenzustellen, sowie zur Laufzeit durch die RTI die federation zweckgemäß betreiben zu können.

5.4.2 Verwaltung einer Federation - Aufgaben der runtime infrastructure

HLA-*federations* sind typischerweise verteilte Systeme. Daher laufen *federates* häufig auf unterschiedlichen Rechnern, was die üblichen Probleme beim Betrieb und der Verwaltung solcher System nach sich zieht [120]. Hieraus ergeben sich für die RTI zwei zentrale Aufgaben. Zum einen ist die Verwaltung einer verteilt ablaufenden Simulation zu koordinieren. Zum anderen sind zentrale Dienste bereitzustellen, die bei der Simulation häufig verwendet werden. Hierzu gehören

- **Organisation der Federation:** Durch diesen Dienst wird zum einen die *federation* durch Vorhandensein und Zugehörigkeit von *federates* beschrieben und andererseits eine Reihe Operationen festgelegt, die innerhalb der gesamten *federation* gültig sind und für deren Koordination notwendig sind, beispielsweise das Bereitstellen von Synchronisationspunkten.
- **Verwaltung von Daten- und Informationsfluss:** Diese Dienste organisieren den Informationsfluss, d.h. den Austausch von Informationen, innerhalb einer *federation*. Grundsätzlich findet ein solcher Austausch (Parameter, Ereignisse) durch einen publisher-subscriber Prozess statt. Hierbei legt jeder *federate* fest, welche Daten er erzeugt bzw. benötigt. *Federates* senden hierbei direkt keine Informationen zu anderen *Federates*, sondern deklarieren zunächst bereitgestellte oder benötigte Informationen. Zur Laufzeit ist es dann die Aufgabe der RTI, die daraus entstehenden Informationsströme zu koordinieren, d.h. diese Verwaltungsdienste bilden die Grundlage der RTI für die Verteilung und Transformation von Daten.
- **Verwaltung der Zuständigkeit / Eigentümerschaft:** In einer verteilten Simulationsumgebung, wie sie HLA anstrebt, sind einzelne Parameter (Attribute) nicht zwangsläufig an einen *federate* gebunden. Es wird vielmehr davon ausgegangen, dass Attribute durch zuständige *federates* verwaltet werden, jedoch diese Zuständigkeit nicht dauerhaft festgelegt werden kann. Entsprechend wird vorgesehen, dass Attribut-Zuständigkeiten innerhalb einer *federation* geteilt oder verschoben werden können. Die Gültigkeit von Zugriffen wird von der RTI überwacht und entsprechend der Verwaltungsvorgaben geregelt.
- **Verwaltung der Zeit:** Ein zentraler Punkt beim Betrieb von verteilten Anwendungen - hier Simulationsanwendungen - ist die Verwaltung der Zeit. Da davon ausgegangen werden muss, dass jeder *federate* auf einem eigenen Rechner abläuft, wäre der zeitliche Fortschritt innerhalb einer *federation* ohne Koordination uneinheitlich. Da die *federation* eine Aufgabe gemeinsam erfüllen soll, muss jedoch ein organisierter Daten- und Ereignisaustausch stattfinden, der sicherstellt, dass

Nachrichten beim Empfänger vor dem Zeitpunkt eintreffen, zu dem die entsprechende Reaktion Effekt zeigen soll. Erreicht die Nachricht den Empfänger zu spät, sind alle Operationen, die nach dem Gültigkeitszeitpunkt der Nachricht durchführt wurden, ungültig. Da zwischenzeitlich u.U. vom Empfänger weitere Nachrichten verschickt wurden, ist in einem solchen Fall die Konsistenz der gesamten *federation* in Frage gestellt. Zur Vermeidung dieser Probleme stellt HLA einen Dienst bereit, der die zeitliche Synchronisation innerhalb einer *federation* regelt.

Wie an dieser Aufzählung zu erkennen, charakterisieren diese Dienste eine Funktionalität im Rahmen eines Frameworks zur verteilten Ausführung komplexer Simulationsszenarien.

5.4.3 Stand der HLA-Entwicklung

Derzeit wird HLA innerhalb von zwei Gremien als Standardisierungsvorschlag diskutiert. 1998 ist die Schnittstellendefinition der HLA durch die Object Management Group (s.o.) verabschiedet worden [121]. Parallel dazu liegt ein Standardisierungsvorschlag der IEEE vor [122].

Obwohl HLA als Spezifikation bereits weit entwickelt ist, konnte sich diese Architektur nur erst ansatzweise im militärischen Bereich [123] sowie in den Forschungsarbeiten einiger Hochschulen [124] durchsetzen. Eine nennenswerte Akzeptanz seitens industrieller Anwender und besonders bei Herstellern marktgängiger Simulationssysteme kann bisher nicht verzeichnet werden, wie aktuelle Konferenzen [125,126,127] und eine Recherche unter namhaften Hersteller von Simulationssoftware auf der Basis von [141] zeigen.

5.5 Auswahl von Architektur und Werkzeugen

Die folgende Bewertung der Architekturen soll untersuchen, inwiefern diese für die Implementierung der eingangs beschriebenen Ziele geeignet sind. Dazu ist zwischen den Middlewares DCOM und CORBA, sowie dem Framework HLA zu unterscheiden. Der Grund, HLA trotzdem gemeinsam mit DCOM und CORBA zu besprechen liegt in den konzeptionellen Anforderungen des Frameworks für verteilte Simulationswendungen (HLA) an eine generische Middleware.

Der Einsatz von DCOM ist häufig durch die verbreitete Entwicklungsplattform 'Windows' motiviert. Für den Entwickler ist es daher sehr hilfreich, auf eine Vielzahl fertiger ('off-the-shelf') Komponenten zurückgreifen zu können. Diese können bei der Applikationsentwicklung dank guter Entwicklungswerkzeuge meist einfach in eine

Windowsumgebung integriert werden. Der Vorteil liegt also in einer kostengünstigen Realisierung des Zielsystems.

Die Nachteile von DCOM liegen in seiner proprietären, microsoftspezifischen Architektur. Der de-facto Industriestandard ergibt sich daher nicht zwangsläufig aus der technischen Leistungsfähigkeit, sondern erscheint eher als Folge der Akzeptanz von Windows als Betriebssystem. DCOM ist dem entsprechend plattformabhängiger. Zudem zeigen neue Microsoftentwicklungen, dass DCOM derzeit nicht zu den Produkten der neuen Unternehmensstrategie Dotnet kompatibel ist und auch keine Migrationswerkzeuge von Microsoft bereitgestellt werden [128].

Im Gegensatz dazu unterstützt CORBA deutlich mehr Plattformen. Dies zeigt sich beispielsweise daran, dass im Sinne einer Middleware der Einsatz von CORBA weitgehend systemunabhängig ist. Ebenso ist CORBA verglichen mit DCOM der offenere Standard, der von mehreren hundert Mitgliedern der OMG getragen wird (darunter auch Microsoft). Die Offenheit wird u.a. durch mehrere standardisierte Abbildungen von der Schnittstellenbeschreibungssprache IDL auf Implementierungssprachen unterstrichen, wozu es -selbst in einem umfangreicheren Vergleich weiterer Middlewarearchitekturen - keine Parallele gibt. Speziell die CORBA-Unterstützung seit Version 1.2. [129] der Java-Plattform wird hier kurz- und mittelfristig positive Effekte zeigen.

Darüber hinaus weist eine umfangreiche Bewertung von [130] auf weitere Schwächen von DCOM hin. Neben den bereits genannten Produkteigenschaften wird hier besonders auf eine erheblich schwierigere Skalierung umfangreicher, verteilter Anwendungen, eine fehlende sprechende und hierarchische Namensgebung (URL-Naming) und fehlende persistente Objektreferenzen hingewiesen. Die Geschwindigkeit beider Middlewares unterscheidet sich nicht signifikant.

Aus den genannten konzeptionellen Erwägungen ist daher CORBA die zu bevorzugende Middleware. Besonders unter Berücksichtigung des Ziels, ein informationslogistisches Architekturkonzept zu entwickeln, welches auf einer Vielzahl unterschiedlicher Informationsquellen in einem Unternehmen aufsetzen soll, müsste die Wahl von DCOM als kritisch angesehen werden. Dennoch darf DCOM als Weg zu bestehenden Anwendungen nicht abgelehnt werden. Dies kann durch sogenannte Bridges sichergestellt werden, die fallweise als Mittler zwischen CORBA und DCOM eingesetzt werden können [131]. Sie schaffen somit die technischen Voraussetzungen, um die Existenz von Middlewaregrenzen vor Clients und Servern aufzulösen (Abb. 47), d.h. aus Anwendersicht transparent zu machen, ohne jedoch gleichzeitig die Sicherheits- und Geheimhaltungsaspekte einer solchen Kommunikation zu vernachlässigen.

Speziell die Plattformunabhängigkeit von CORBA war auch der Grund, dass *die runtime-infrastructure* als zentraler Bestandteil von HLA auf CORBA zurückgreift und

viele Referenzimplementationen von HLA diese Middleware nutzen. Insofern kann ein ORB als ein Mittler zwischen verteilten Objekten betrachtet werden und analog dazu die RTI als Mittler zwischen *federates*. Die Unterschiede ergeben sich aus den unterschiedlichen Aufgaben, die CORBA als Middleware und HLA als Framework erfüllen. Als Beispiel sei hier die implizite Interaktion zwischen HLA *federates* auf der einen Seite und den weitgehend expliziten Interaktionen innerhalb eines mit CORBA erstellt Systems zu nennen.

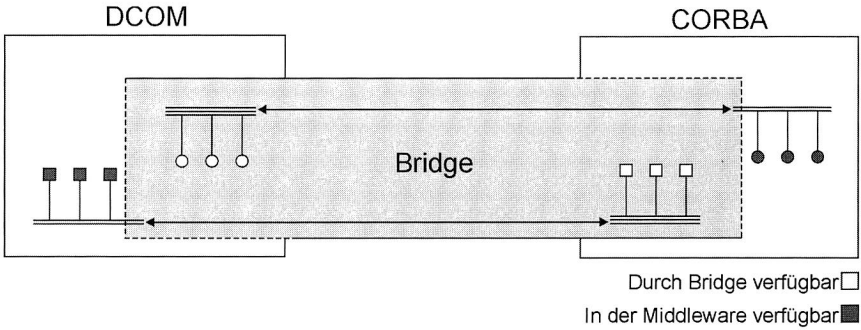


Abb. 47: Verbindung der CORBA- und DCOM-Welt durch eine Bridge

Neuere Forschungsansätze im Zusammenhang mit HLA richten sich hierbei in erster Linie auf eine Standardisierung der Schnittstellen, der Ausführungsumgebung bei der Kommunikation zwischen Simulationskomponenten und zusätzlicher (Meta-) Daten zur Modellbildung innerhalb einer Federation (etwa durch Typinformationen) [132, 133, 134, 135].

In Bezug auf die Datenbeschaffungsproblematik bei der Simulation von Produktionssystemen muss nun eine Abwägung der beiden potentiellen Middleware-Architekturen in der Weise erfolgen, dass die zu konzipierende informationslogistische Architektur ein Maximum an Datenquellen (auf unterschiedlichen Plattformen) in das Konzept problemlos einbinden lässt. In diesem Sinne ist CORBA die geeignete Wahl, wie auch eine ausführliche Bewertung der Middlewares HLA, CORBA und der javabasierten RMI belegt [136].

Gleichzeitig ist abzuwägen, wie der Integrationsaufwand in bestehende Informationslogistiksysteme möglichst gering zu halten ist. Demgemäss wäre eine Windows-basierende Architektur sinnfällig, da dieses System den weitaus höchsten industriellen Durchdringungsgrad aufweist. Zusätzlich zeigen aktuelle Entwicklungen, z.B. OPC (OLE für Process Control), dass Windows zunehmend in operativen Produktionsbereichen erfolgreich eingesetzt werden kann.

Hinsichtlich der in Kap. 4 beschriebenen konzeptionellen Arbeiten waren folgende Argumente entscheidend, CORBA zu verwenden:

- neutraler Standard, einschließlich einer leistungsfähigen Schnittstellenbeschreibungssprache, der interface definition language (IDL)
- als Middleware technisch breiter und offener angelegt
- besser skalierbar
- Bridges erschließen andere Middleware-Welten, z.B. (D)COM, damit können die Vorzüge von DCOM hinsichtlich der Erreichbarkeit von Daten/ Informationen/ Parametern ebenfalls genutzt werden.
- Notwendige Entwicklungswerkzeuge sind in ausreichender Anzahl und Leistungsfähigkeit für alle gängigen Betriebssysteme verfügbar.

Somit ergibt sich für das zu realisierende Architekturkonzept seitens der genutzten Architekturen die Middleware CORBA als Entwicklungskern, von dem ausgehend die gemeinsame Nutzung und Weiterentwicklung hinsichtlich DCOM (Windows/ Microsoft), HLA (als potentieller Standard verteilter Simulationen) sowie weiterer Middlewarekonzepte (besonders neuere JAVA-basierte Entwicklungen) nicht negativ beeinflusst wird.

6 Objektorientierter Architekturentwurf

In Kap. 4 wurden bereits die Anforderungen an ein Kommunikationsframework zur Beschaffung von simulationsrelevanten Daten definiert. Außerdem wurden die in dieser Architektur agierenden Module identifiziert. Im folgenden wird hieraus ein Objektmodell erstellt, indem Schnittstellen (Interfaces) definiert werden, über die der Nachrichtenaustausch zwischen den einzelnen Modulen erfolgt. Nach der Definition der Interfaces erfolgt die Betrachtung der Interaktionen, die hierüber möglich sind.

6.1 Grundlegende Interfaces

Um eine sinnvolle Strukturierung der erstellten Interfaces zu erlauben, wurde eine Vererbungshierarchie gebildet, welche auf Interfaces basiert, die für die implementierenden Objekte eine gemeinsame Grundfunktionalität spezifizieren.

6.1.1 Das Interface basic_object

An der Wurzel der Vererbungshierarchie für die Interfaces der Objekte der Anwendungsschicht befindet sich das Interface *basic_object* (Abb. 48). Es wurde eingeführt, um die Grundfunktionalität aller Objekte zu spezifizieren.

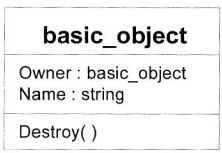


Abb. 48: Das Interface basic_object [1]

Objekte, die dieses Interface implementieren, repräsentieren benannte Einheiten (Entitäten). Um die Bildung von Hierarchien zu ermöglichen, muss zusätzlich die Herkunft, bzw. der Eigentümer des Objekts gespeichert werden. Durch dieses Attribut erhalten Objekte mit dem Interface *basic_object* das Wissen, von wem sie erzeugt wurden. Es wird bei der Erzeugung eines Objekts festgelegt. Es wird weiterhin gefordert, dass diese Beziehung auch dem erzeugenden Objekt bekannt ist. Es muss ebenfalls „wissen", welche Objekte von ihm erzeugt wurden und ihm dadurch zu-/untergeordnet sind. So entsteht letztendlich eine hierarchisch geordnete Baumstruktur. Verwendet werden solche Bäume aus Objekten des Interfaces *basic_object*

[1] Die im folgenden verwendete OMT (object modeling technique) Notation ist in [101] definiert.

für die Darstellung der Hierarchien einer Organisationsstruktur, aber auch, um die Verantwortlichkeit für die Vernichtung eines basic_objects festzulegen.

6.1.2 Das Interface link_object

Das Interface *link_object* (Abb. 49) wird vom Interface *basic_object* abgeleitet und spezifiziert das gemeinsame Grundverhalten aller Objekte, die für die Darstellung der Material- und Informationsflussverbindungen einer Organisationsstruktur der Fertigung benutzt werden.

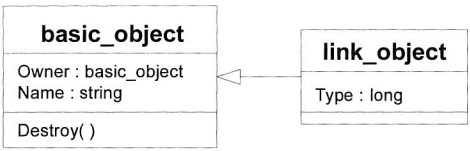


Abb. 49: Das Interface link_object

In Erweiterung von *basic_object* wird durch einen *Type* eine eindeutige Identifikation der dargestellten Verbindung vorgenommen. Die Festlegung, durch welchen Wert von *Type* welche Verbindung zu repräsentieren ist, erfolgt nicht durch das Interface, sondern durch die Werkzeuge die sich seiner Objekte bedienen, um Informationen zu übertragen. Hier muss eine Abstimmung zwischen dem Erzeuger und dem Klienten eines *link_objects* erfolgen. Dem Erzeuger muss bekannt sein, welcher Fluss durch welchen *Type* zu repräsentieren ist. Beim Klienten muss ebenfalls genau definiert werden, wie die Werte von *Type* zu interpretieren sind, um die Verbindung mit seinen Mitteln zu rekonstruieren (siehe auch Kap. 4.7). Dies geschieht durch den im folgenden beschriebenen datatypeContainer, durch den als Metainformation eine Aussage über die auf einem link_object übertragenen Informationen erfolgt.

6.1.3 Das Interface datatypeContainer

Das Interface *datatypeContainer* (Abb. 50) spezifiziert die Grundfunktionalität, die für die einheitliche Repräsentation der simulationsrelevanten Daten benötigt wird. Es ermöglicht Klienten den einheitlichen Zugang zu Informationen unterschiedlichen Typs (Polymorphie).

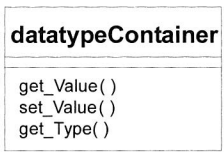


Abb. 50: Das Interface datatypeContainer

Das polymorphe Verhalten des Interfaces *datatypeContainer* wird erreicht, indem es für jeden repräsentierbaren Datentyp eine Implementierung besitzt. Objekte, die dieses Interface implementieren, werden von ihren Klienten wie Attribute benutzt. Sie speichern Informationen in ihnen und können diese, falls erforderlich, wieder aus ihnen extrahieren. Aus diesem Grund werden die Objekte, die das Interface *datatypeContainer* implementieren im weiteren Verlauf als Attributobjekte bezeichnet.

Die verwendeten Datentypen orientieren sich an den physikalischen Einheiten, die aus der korrekten Transformation einer Zahl zu einem Parameter der Datenklasse S entstehen, sowie der internen Repräsentation der verwendeten Schnittstellenbeschreibungssprache IDL (Tabelle 3).

Interpretation der Struktur als...	Bezeichnung des *datatype*-Elements	IDL-Typ des im *value*-Element gespeicherten Wertes
Wahrheitswert	**BOOLEAN**	**boolean**
Zeichenkette	**STRING**	**string**
Ganze Zahl	**LONG**	**long**
Gleitpunktzahl	**DOUBLE**	**double**
Länge in [m]	**LENGTH**	**double**
Gewicht in [kg]	**WEIGHT**	**double**
Zeit in [s]	**TIME**	**double**
Datum	**DATE**	**string**
Geschwindigkeit in [m/s]	**SPEED**	**double**

Tabelle 3: Verwendete Datentypen im datatypeContainer

Da das Interface *datatypeContainer* nicht vom Interface *basic_object* abgeleitet wurde, besitzen seine Objekte keinen Namen und können auch nicht für die Hierarchiebildung benutzt werden. Sie sind nur für das Objekt sichtbar, von dem sie erstellt wurden und können in ihm über einen Namen oder einen Index eindeutig identifiziert werden. Wie diese Namen und Indizes verwaltet werden, ist Sache der Implementierung und wird nicht durch das Interface festgelegt. Andere Objekte können nur über das Interface des besitzenden Objekts auf sie zugreifen.

Das Interface selbst benötigt Operationen, die es dem besitzenden Objekt ermöglichen, auf die repräsentierten Informationen zuzugreifen, sowie Operationen um zwischen Informationen unterschiedlichen Typs zu unterscheiden. Entsprechende Ope-

rationen sind vorgesehen, um lesend und schreibend auf das Objekt zuzugreifen, sowie den Type als Metainformation abfragen zu können.

6.2 Interfaces für die Verbindung der Simulation mit der Datenschicht

6.2.1 Das Interface proxy

Für die Klasse der Proxyobjekte (sie Kap. 4.5) wurde das Interface *proxy* modelliert (Abb. 51). Die *Proxys* sind die Repräsentanten der eigentlichen Kommunikationsobjekte. Sie verfügen über den gleichen Datenbestand wie diese. Aus diesem Grund kann die Simulation durch die Benutzung eines *Proxys* auf die aktuellen Daten des originären Kommunikationsobjekts zugreifen.

Abb. 51: Das Interface proxy

Um bei dieser doppelten Datenhaltung Inkonsistenzen zu vermeiden, enthält das *proxy*-Interface entsprechende Funktionen, mit denen der Datenabgleich zwischen Kommunikationsobjekt und Proxy durchgeführt wird. Sie werden vom originären Kommunikationsobjekt aufgerufen, nachdem die in ihm enthaltenen Daten verändert wurden.

Um es der Simulation zu ermöglichen, die simulationsrelevanten Daten aus der Datenschicht zu akquirieren, wird der Proxy zunächst initialisiert, d.h. die aktuellen Daten werden aus der Datenschicht beschafft und in den Attributobjekten des *Proxys* hinterlegt. Dies geschieht, indem der *Proxy* die Initialisierung des originären Kommunikationsobjekts anstößt. ier erfolgt dann die eigentliche Datenakquisition. Die vom Kommunikationsobjekt beschafften Daten werden anschließend durch die Operation *update* in den *Proxy* übertragen.

Die bei der Initialisierung stattfindenden Vorgänge werden bei der Beschreibung der Initialisierungsfunktion der Kommunikationsobjekte erläutert (6.2.2), da im *Proxy* nur eine Delegation an diese erfolgt. Der Grund hierfür liegt in der gewählten dreischichtigen Architektur, bei der die funktionalen Belange innerhalb der Anwendungsschicht behandelt werden (das Interface Proxy ist demzufolge der Darstellungsschicht zuzuordnen).

Nach der Initialisierung und nachdem die benötigten Daten in den Attributobjekten hinterlegt worden sind, werden Operationen benötigt, die es der Simulation erlauben, die von ihr benötigten Daten (erneut) anzufordern. Hierbei ist vorgesehen, dass einzelne Attributobjekte gezielt angesprochen werden können. Dies hat den Vorteil, dass kein zeitaufwendiger, kompletter Datenabgleich zwischen den Schichten stattfinden muss, wenn dies nicht notwendig ist.

Weiterhin ist simulationsseitig neben dem lesenden auch ein schreibender Zugriff auf die Attributobjekte zu ermöglichen. Dies ist notwendig, um einen Proxy für den Datentransport aus einem Simulationsmodell heraus nutzen zu können. Die konsequente Nutzung dieses Ansatzes führt dazu, dass das Proxy-Interface über Operationen verfügt, die es einem Klienten (der Simulation) erlauben, weitere Attributobjekte zu erzeugen und zu vernichten. Demzufolge ist so die Möglichkeit gegeben, den genauen Datenumfang innerhalb eines Simulationsmodells zu spezifizieren, welcher über ein Proxy- und Kommunikationsobjekt akquiriert oder bereitgestellt wird.

Das Interface benötigt darüber hinaus Operationen, um die (asynchronen) Ereignismeldungen, die vom Kommunikationsobjekt stammen, an die Simulation weiterzuleiten. Die Operationen, die das Interface dafür bereitstellt, werden einerseits vom Kommunikationsobjekt benutzt, um eine von ihm empfangene Ereignismeldung an den *Proxy* weiterzuleiten und anderseits von der Simulation, um die Zustellung einer Meldung zu unterbinden (zu maskieren) oder zuzulassen (zu demaskieren).

Wie die Weiterverarbeitung einer empfangenen Ereignismeldung geschieht, wird durch die spezifische Implementierung festgelegt und hängt vom Simulationswerkzeug und den in ihm enthaltenen Strukturen ab. Diese Tatsache hat zur Folge, dass Proxyobjekte für einen Simulator „maßgeschneidert“ werden müssen.

Zuletzt ist innerhalb dieses Interfaces eine Schnittstelle vorgesehen, durch die die Simulation die Steuerung der Systeme der Datenschicht vornehmen kann, d.h. Operationen im Datenobjekt ausführen kann. Ebenso wie die Initialisierung wird auch diese Operation an ein Kommunikationsobjekt delegiert.

6.2.2 Das Interface Kommunikationsobjekt: comobject

Das Interface *comobject* (Abb. 52) wurde für die Objekte modelliert, die bei der Erstellung des Konzepts als Kommunikationsobjekte (4.6.2) bezeichnet wurden. Diese Objekte akquirieren Daten aus der Datenschicht und speichern diese in ihren Attributobjekten ab. Von hier aus können sie von den Klienten erreicht werden. Das Interface benötigt daher Operationen, die die Akquisition der Daten ermöglichen, d.h. mit den Datenobjekten (wrappern) interagieren, und Operationen, mit denen diese Daten den Klienten zugänglich gemacht werden.

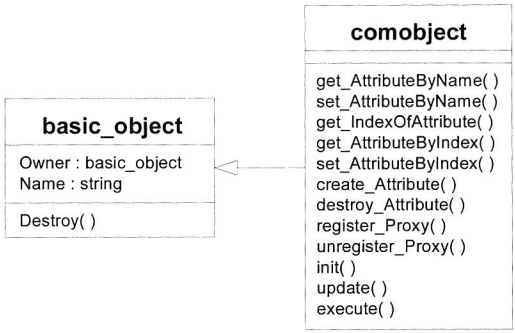


Abb. 52: Das Interface comobject

Die Akquisition der Daten geschieht über Operationen des Interfaces, welche die Initialisierung des Objekts einleiten, über die entsprechenden Wrapper die benötigten Daten aus der Datenschicht beschaffen und diese in die entsprechenden Attributobjekte übertragen.

Bei der Initialisierung soll gemäß der Anforderung auch eine Zugriffsstrategie für die robuste Datenakquisition festgelegt werden können. Durch sie wird die Vorgehensweise bei der Suche nach Datenquellen während der Initialisierung und allen späteren Zugriffen auf die Datenquelle festgelegt.

Die anwendbaren Strategien beschreiben entweder eine Reihenfolge, in der auf die relevanten Datenquellen abhängig von ihrer Verfügbarkeit zugegriffen wird, oder geben eindeutig an, welche Datenquelle zu benutzen ist (Tabelle 4). Falls an dieser Stelle die Anbindung an eine reale Fertigungskomponente stattfindet, kann zusätzlich eine Verbindung mit dem Nachrichtendienst (*Event Service)* erfolgen, da dieser für die Zustellung von Störinformationen, d.h. asynchronen Ereignissen, benutzt wird. Tabelle 4 veranschaulicht, in welcher Reihenfolge die Anbindung an die vorhandenen Datenquellen versucht wird, und ob eine Anbindung an den *Event Service* stattfindet.

Eine diesbezüglich noch aufzugreifende Teilaufgabe ist der Datenabgleich zwischen 'realen' Datenquellen und einer Datenbank. Dies wird als eigenständiges Subsystem im Rahmen zusätzlicher Dienste für die Architektur in Kap. 6.7 behandelt.

Regel	1. Datenquelle	2. Datenquelle	Anbindung an den *Event Service*
1	Maschine	——	ja
2	Maschine	Datenbank	ja (wenn Anbindung eine Maschine)
3	Datenbank	——	nein
4	Datenbank	Maschine	ja (wenn Anbindung eine Maschine)
5	frei definierbar		benutzerdefiniert

Tabelle 4: Strategien für die Initialisierung eines Kommunikationsobjekts

Die flexibelste fünfte Regel sieht keine direkte Selektion zwischen zwei Datenquellen vor, sondern arbeitet über einen weiteren Verweis auf eine virtuelle Datenquelle, die ihrerseits komplexe Zugriffsregeln beinhaltet. Damit kann die Organisation der Datenbeschaffung delegiert und/oder weiter im Gesamtsystem verteilt werden. Ebenso wird hierdurch beispielsweise die Nutzung von Expertensystemen ermöglicht, die bedarfsweise Expertenwissen in die Akquisition von Informationen einbringen. Diese Möglichkeit wurde von [69] bereits erfolgreich für die Bildung von Simulationsmodellen eingesetzt. Neuere Arbeiten zeigen hier ebenfalls erfolgversprechende Potentiale bei der Datenaufbereitung auf [150].

Nachdem das Objekt erfolgreich initialisiert wurde und die benötigten Daten in seinen Attributobjekten hinterlegt sind, werden Operationen benötigt, die es einem Klienten erlauben, die von ihm benötigten Daten anzufordern. Die gesamte Verwaltung der Attributobjekte entspricht funktional der von Proxys.

Um sicherzustellen, dass die Daten der Proxyobjekte und die des Kommunikationsobjekts konsistent bleiben, wird ein Mechanismus genutzt, der bei einer Veränderung der Daten die erforderlichen Aktualisierungen automatisch bei allen registrierten Proxys durchführt. Dies geschieht über das *publish-and-subscribe* Verfahren (Abb. 53). Hierbei können Objekte (*subscriber*), die regelmäßig die Informationen eines anderen Objektes (*publisher*) benötigen, diese 'abonnieren', indem sie ihre Referenz hinterlegen. Dieser Vorgang wird auch Registrierung genannt. Anschließend kann der *publisher*, wann immer es nötig ist, dem registrierten *subscriber* die benötigten Informationen zukommen lassen. Um es einem *Proxy* zu ermöglichen, nach der Änderung eines seiner Attributobjekte (z.B. bei einem schreibenden Zugriff durch die Simulation) das entsprechende Attributobjekt des Kommunikationsobjekts zu aktualisieren, können die Attributobjekte eines Kommunikationsobjekts auch durch einen Proxy verändert werden.

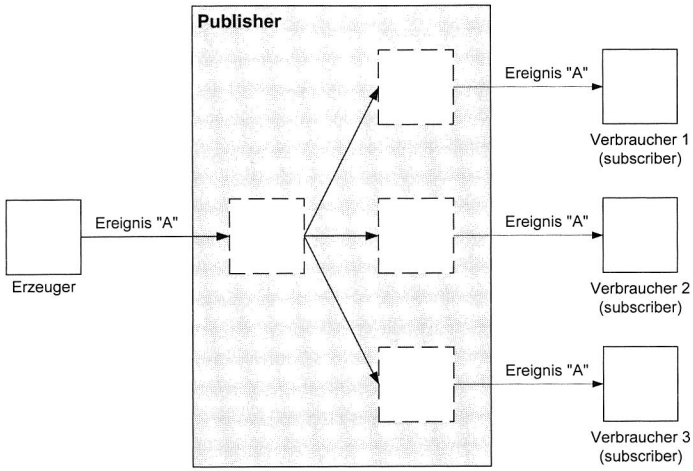


Abb. 53: Grundprinzip des publish-and-subscribe Verfahrens zur Verteilung von Ereignissen an mehrere Verbraucher ('Abonnenten')

Alle weiteren Methoden dieses Interfaces arbeiten analog zur Beschreibung des Proxy-Interfaces, da die dort aufgeführten Operationen letztlich an das Kommunikationsobjekt delegiert werden.

6.2.3 Das Interface wrapper

Um die Systeme der Datenschicht in die Architektur zu integrieren, wurde ein Interface modelliert, das die Realisierung der in Kap. 4.6.1 vorgestellten Wrapper ermöglicht. Dieses Interface erlaubt durch seine Operationen den Zugriff auf das gekapselte Datenschicht-System.

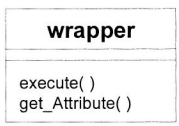


Abb. 54: Das Interface wrapper

Die Operationen des *wrapper*-Interfaces (Abb. 54) werden von den Kommunikationsobjekten benutzt, um die Dienste und Daten der in ihm gekapselten Systeme zu verwenden. Um dies zu ermöglichen, muss das *wrapper*-Interface folgenden Anforderungen gerecht werden:

1. Der Zugriff auf die simulationsrelevanten Informationen des gekapselten Systems muss möglich sein. Dies wird verlangt, um während der Initialisierung an die von der Simulation benötigten Daten zu gelangen.

2. Es muss beim gekapselten System die Ausführung einer Aktion veranlassen können.

Das Verhalten des wrappers hängt in hohem Maße vom gekapselten System ab. Bei einer Maschine z.B. ist es ausreichend, wenn der Name der verlangten Kenngröße angegeben wird. Handelt es sich dabei jedoch um eine Datenbank, die Informationen über mehrere Fertigungskomponenten verwaltet, ist eine eindeutige Identifizierung über den Namen der Kenngröße nicht möglich. Es wird hier auch der Name der Fertigungskomponente benötigt, auf die sich die Informationen beziehen.

Die durch das *wrapper*-Interface gekapselten Systeme des Produktionssystems sind auch die Quellen der durch die Architektur weitergeleiteten Ereignismeldungen. Da für diese Aufgabe der *Event Service* gewählt wurde, und die Anbindung an ihn nicht über das Interface, sondern über die Implementierung (an das zugeordnete Kommunikationsobjekt) erfolgt, besitzt das *wrapper*-Interface keine Operationen für die Übermittlung von Ereignismeldungen.

Die Vielfalt der potentiell nutzbaren Datenquellen bedingt weitere Überlegungen, wie mit der hier anzunehmenden Verschiedenheit umzugehen ist. Hierdurch ist auch die geringe Operationenanzahl dieses Interfaces zu erklären - Daten werden über ihren Namen angesprochen, die eigentliche Zuordnung von Name zu tatsächlichem Zugriff auf die Interna der Datenquelle erfolgt i.S. einer Interpretation im wrapper. Darüber hinaus ermöglichen leistungsfähige CORBA-Implementationen bis auf Feldbusebene einen direkten Zugriff auf Prozessdaten, wie Arbeiten in [137,138,139,140] zeigen. Im Betrieb sind die vorhandenen Attribute dabei entweder bekannt oder werden für die Operation *Execute ("ListAttributes")* ermittelt.

6.3 Interfaces für die Repräsentation von Organisationsstrukturen

Für die Darstellung von Strukturen der Fertigung, die neben der eigentlichen Kommunikationsanbindung die zweite Aufgabe der Anwendungsschicht ist, wurden in 4.5.3 drei Klassen vorgestellt. Diese drei Klassen, die Simulationsobjekte, die Netzwerkobjekte und die Modellobjekte, repräsentieren Organisationseinheiten und können Beziehungen eingehen, die es ermöglichen, eine beliebige hierarchische Organisationsstruktur zu repräsentieren. Es wurde auch erwähnt, dass eventuell vorhandene Informations- und Materialflussverbindungen zwischen den Fertigungskomponenten darstellbar sind. Dies geschieht, indem für die obengenannten Objekte Schnittstellen geschaffen werden, durch die ein Informations-/Materialfluss erfolgen kann. Ein tatsächlich stattfindender Fluss wird durch die Verbindung zweier entsprechender Schnittstellen repräsentiert. Diese Informations- und Materialflussschnitt-

stellen sowie ihre Verbindungen werden durch Objekte der Klassen *Inport, Outport* und *Link* repräsentiert. Im weiteren Verlauf werden diese Interfaces als Repräsentanten der Datenklasse S in dieser Architektur aufgefasst und zur Repräsentation von Organisationsstrukturen benutzt.

6.3.1 Das Interface port

Objekte mit dem Interface *port* repräsentieren abstrakte Schnittstellen einer Organisationseinheit. Durch sie erfolgt ein gerichteter Fluss von Informationen. Dieses Interface ist eine Generalisierung von *link_object* (Abb. 55), da seine Objekte keine Fertigungskomponenten oder aus ihnen gebildete Strukturen repräsentieren, sondern nur die Schnittstellen, die mit den Schnittstellen anderer Objekte Verbindungen eingehen können.

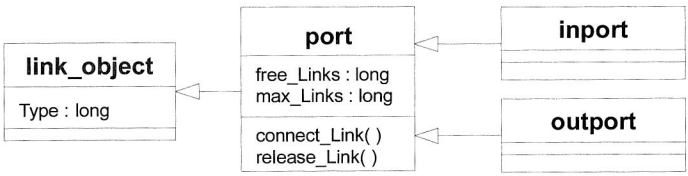


Abb. 55: Die Interfaces port, inport und outport

Der Besitzer der durch ein *port*-Objekt repräsentierten Schnittstelle ist das Objekt, das ihm übergeordnet ist. Seine Referenz befindet sich in einem geerbten *basic_object*. Die Art des Gutes, das über ein *port*-Objekt transportiert wird, wird im vom *link_object* geerbten Attribut *Type* festgelegt. Die Anzahl der Verbindungen kann bedarfsweise beschränkt werden.

Da Material- und Informationsflüsse gerichtet stattfinden, werden zwei unterschiedliche Arten des *port*-Interfaces benötigt. Eines für eingehende und eines für ausgehende Schnittstellen. Dies wird durch die beiden Interfaces *inport* und *outport* realisiert, die zwar vom *port*-Interface abgeleitet sind, dieses aber nicht erweitern. Der Aufbau dieser beiden Interfaces ist strukturell identisch. Sie unterscheiden sich nur in ihrer Semantik.

6.3.2 Das Interface link

Das Interface *link* wird von Objekten implementiert, die eine Verbindung zwischen zwei Organisationseinheiten darstellen. Es verbindet zwei Objekte mit dem Interface *port* und repräsentiert einen gerichteten Material- oder Informationsfluss zwischen diesen. Die Richtung der Verbindung wird durch die beiden Attribute *Source* und *Target* definiert. Im ersten befindet sich die Referenz des *ports,* der den Ursprung des Flusses repräsentiert. Im zweiten befindet sich die Referenz des *ports,* durch

den die Repräsentation des Zieles erfolgt. Die Art des durch einen *link* repräsentierten Flusses wird durch das vom Interface *link_object* (Abb. 56) geerbte Attribute *Type* festgelegt. Es muss den gleichen Wert wie die *Type*-Attribute der beiden verbundenen *ports* haben, da ein *link* nicht die Umwandlung, sondern ausschließlich den Transport eines Gutes repräsentiert.

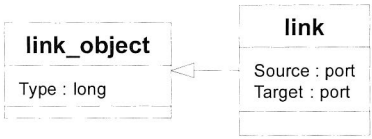


Abb. 56: Das Interface link

Über das geerbte *basic_object* können *links* in eine Hierarchie eingeordnet werden. Das ihnen übergeordnete Objekt repräsentiert das (Teil-) System, in dem sich die durch ein *link*-Objekt repräsentierte Verbindung befindet. Um hier eine eindeutige Zuordnung zu ermöglichen, dürfen Verbindungen nicht über die Grenzen eines (Teil-) Systems hinaus gehen, d.h. die Objekte die verbunden werden, müssen Komponenten des gleichen (Teil-)Systems repräsentieren. Aus diesem Grund können *link*-Objekte nur drei Arten von Verbindungen eingehen:

Verbindungen zwischen den Komponenten eines (Teil-)Systems: Sie verbinden die *ports* zweier Objekte, die Komponenten des gleichen Teilsystems repräsentieren. Dabei sind nur *links* erlaubt, die von einem *outport* ausgehen und zu einem *inport* führen, siehe auch Abb. 57.

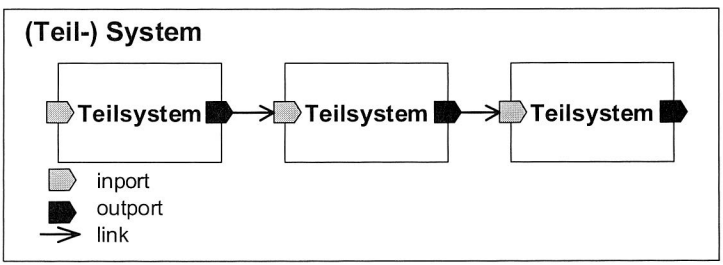


Abb. 57: Verbindungen der Komponenten eines (Teil-)Systems

Ankommende Material- und Informationsflüsse: Sie erfolgen über die ins Teilsystem führenden Schnittstellen, also den *inports* eines Objekts, das eine funktionale oder logische Einheit repräsentiert (Abb. 58). Die Weiterführung dieser Flüsse an die Komponenten des Teilsystems erfolgt durch *links* zwischen den *inports* des Objekts, das das Teilsystem repräsentiert und den *inports* der Objekte, mit denen die Komponenten des Teilsystems repräsentiert werden.

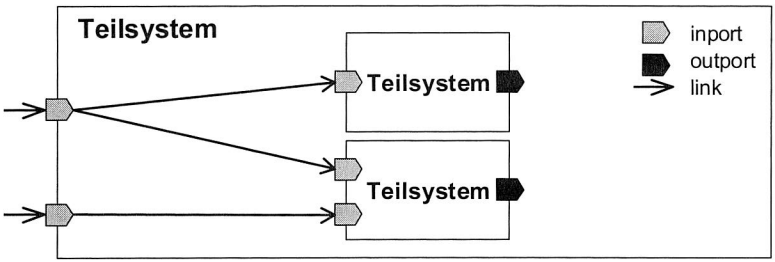


Abb. 58: Ankommende Material- und Informationsflüsse

Abgehende Material- und Informationsflüsse: Sie erfolgen von den Komponenten eines Teilsystems zu den aus dem Teilsystem führenden Schnittstellen (Abb. 59). Die Repräsentation solcher Verbindungen erfolgt durch *links,* die von den *outports* der Objekte mit denen die Komponenten des Teilsystems repräsentiert werden, zu den *outports* des Objekts führen, das das Teilsystem repräsentiert.

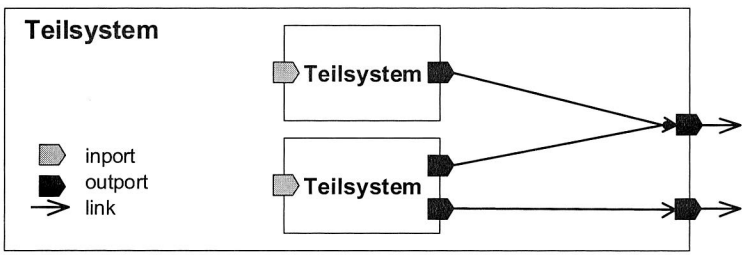


Abb. 59: Abgehende Material- und Informationsflüsse

6.3.3 Das Interface simobject

Für die Simulationsobjekte aus Kapitel 4 – durch sie erfolgt die Repräsentation der atomaren Einheiten einer Organisationsstruktur – wurde das Interface *simobject* modelliert. Es ist eine Generalisierung des Interfaces *comobject* (Abb. 60) und ermöglicht dadurch, dass die Objekte, die *simobject* implementieren, auch für die Kommunikation mit den realen Fertigungskomponenten benutzt werden können.

Durch ein Identifikationsattribut wird eindeutig festgelegt, von welcher Art die durch ein Simulationsobjekt repräsentierte Fertigungskomponente ist. Dieses Attribut ordnet jedem Simulationsobjekt eine Fertigungskomponente zu. Dabei muss der Erzeuger eines Simulationsobjekts wissen, welche *ID* welcher Fertigungskomponente zugeordnet ist. Diese eindeutigen Identifikationen werden durch die Verwaltungsinstanzen der Hierarchien vorgenommen.

Die Simulation hingegen muss weiterhin eine eindeutige funktionale Zuordnung zwischen der Identifikation des Objekts (Struktur) und eines Simulationsobjekts (Dynamik) herstellen können. Dazu benötigt sie z.B. eine Tabelle oder eine Liste, in der die Zuordnung der Identifikation zu einem Objekt des Simulators erfolgt.

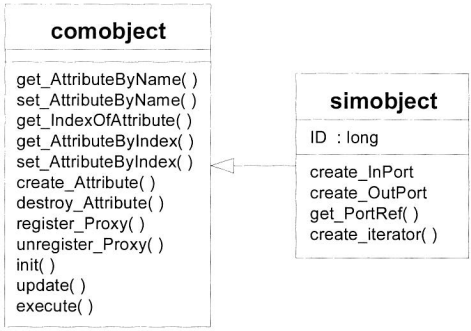


Abb. 60: Das Interface simobject

Das Interface benötigt Operationen, durch die es möglich ist, die in einem Simulationsobjekt vorhandenen Material- und Informationsflussschnittstellen zu erzeugen. Dies geschieht durch die Operationen *create_InPort* und *create_OutPort.* Durch sie werden die benötigten *Port*-Objekte erzeugt und dem Simulationsobjekt untergeordnet.

6.3.4 Das Interface object_iterator

Weiterhin wird Klienten ermöglicht, Informationen über den internen Aufbau eines Simulationsobjekts anzufordern. Hierzu gehören Operationen, durch die ein Klient Zugriff auf die *Port*-Objekte eines Simulationsobjekts erhält. Sie wird z.B. bei der Verbindung der *Ports* eines Simulationsobjekts benötigt. Ebenso ist ein Iterator notwendig, durch den Klienten detaillierte Informationen über den Aufbau eines Simulationsobjekts abfragen können.

Iteratoren sind Objekte, die von einem Klienten eingesetzt werden, um Näheres über den Aufbau eines bestimmten Objekts zu erfahren. Sie erlauben in einer genau festgelegten Reihenfolge den Zugriff auf alle Komponenten eines Objekts [109 . Sie werden vom Objekt erstellt, das sie „auskundschaften" sollen, und können nur Informationen über dieses eine Objekt liefern.

Um es den Klienten der Architektur zu ermöglichen, den Aufbau einer Organisationsstruktur zu erkunden, wurde daher ein Interface entworfen, das es ermöglicht, für die daran beteiligten Objekte Iteratoren zu implementieren. Dieses Interface heißt *object_iterator* und wird vom Interface *basic_object* abgeleitet (Abb. 61). Das ist nötig,

um den Iterator dem Objekt zuzuordnen, für das er erstellt wurde, da dieses für die Erzeugung und die Vernichtung seiner Iteratoren verantwortlich ist.

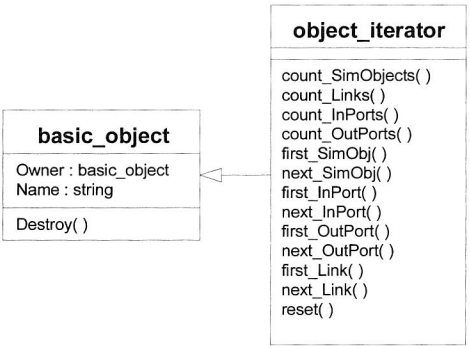


Abb. 61: Das Interface object_iterator

Dem Iterator kommt eine wichtige Bedeutung bei der Generierung von Simulationsmodellen aus der Anwendungsschicht heraus zu. Ausgehend von einer Modellrepräsentation im scratch-board wird der Iterator die dort gespeicherte Struktur durchlaufen und das Erzeugen der entsprechenden Strukturen im Simulationssystem veranlassen. Hierzu ist eine definierte Reihenfolge einzuhalten, da beispielsweise Verbindungen zwischen Komponenten erst dann erzeugt werden können, wenn vorher die Komponenten selbst erzeugt wurden. Dementsprechend ist bei der Erzeugung des Simulationsmodells die Datenstruktur ausgehend von der Wurzel (Modellobjekt) schichtweise zu durchlaufen, wie Abb. 62 verdeutlicht.

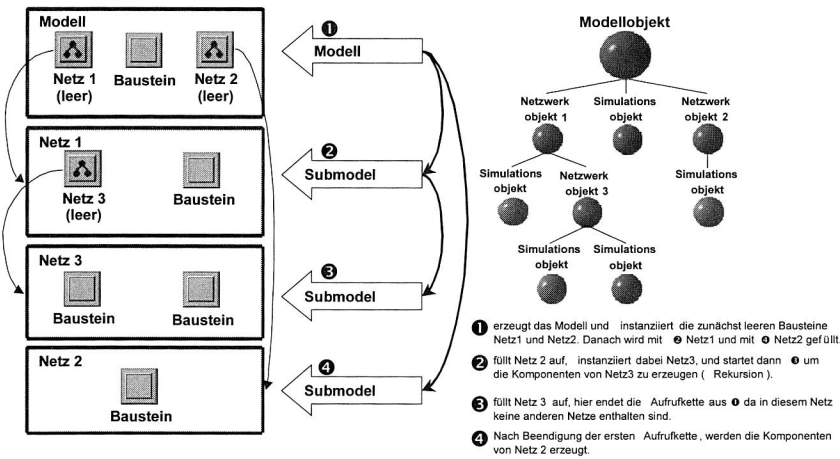


Abb. 62: Einsatz eines object_iterators zur Erzeugung eines Simulationsmodells

Die *count_SimObjects*-Operation des Interfaces liefert die Anzahl der *simobjects,* die dem iterierten Objekt untergeordnet sind. Durch die Operationen *count_InPorts*, *count_OutPorts* und *count_Links* wird dem Klienten die Anzahl der dem iterierten Objekt untergeordneten *inport*-, *outport*- oder *link*-Objekte zurückgegeben. Die Operationen *first_SimObj* und *next_SimObj* liefern dem Klienten den Namen, die Referenz und die *ID* des ersten, beziehungsweise des nächsten dem iterierten Objekt untergeordneten *simobject*. Die Reihenfolge, in der dies geschieht, hängt von der Implementierung ab. Die restlichen mit *first_* und *next_* beginnenden Operationen verhalten sich ähnlich, beziehen sich aber auf die Objekte, die dem weiteren Namen der Operation entsprechen. Durch Ausführen der *reset*-Operation wird der Iterator zurückgesetzt.

6.3.5 Das Interface network

Das Interface *network* wird von den Objekten bereitgestellt, die Organisationseinheiten eines Fertigungssystems (jedoch nicht das gesamte System) hierarchisch zu repräsentieren haben und mit denen andere Organisationseinheiten zu einer logischen oder funktionalen Einheit zusammenfasst werden. Diese untergeordneten Einheiten werden entweder durch Simulationsobjekte oder durch Netzwerkobjekte selbst dargestellt. Aus diesem Grund benötigt das *network*-Interface (Abb. 63) Operationen, die es ermöglichen, obengenannte Objekte zu erzeugen und zu vernichten. Außerdem benötigt es Operationen, mit denen die Objekte erzeugt werden, die für die Repräsentation von Informations- und Materialflussverbindungen nötig sind.

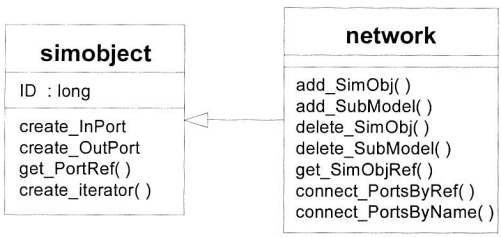


Abb. 63: Das Interface network

Es generalisiert das Interface *simobject*, da alle Eigenschaften von *simobject* auch in *network* benötigt werden. Für die Erzeugung der Komponenten eines Netzwerkobjektes sind Operationen notwendig, die benötigte Simulationsobjekte oder (Unter-) Netzwerke erzeugen. Es ist zu berücksichtigen, dass diese Operationen nur leere Objekte erzeugen. Die in ihnen enthaltenen Strukturen müssen ebenfalls durch einen Klienten verändert werden, was sowohl die Verbindungen zwischen den gespeicherten Objekten als auch deren Vernichtung zu berücksichtigen hat.

6.3.6 Das Interface model

Die oberste Hierarchiestufe und damit das gesamte Fertigungssystem wird durch Modellobjekte repräsentiert. Solche Objekte definieren ein Gesamtsystem und bestehen aus den gleichen Komponenten wie die Netzwerkobjekte, nämlich Netzwerkobjekten und Simulationsobjekten sowie den Verbindungen, die zwischen ihnen bestehen. Außerdem repräsentieren diese beiden Klassen einen ähnlichen Sachverhalt (Systeme die aus Teilsystemen bestehen). Diese Überlegungen legen den Schluss nahe, für die Modellobjekte das gleiche Interface wie für die Netzwerkobjekte zu benutzen.

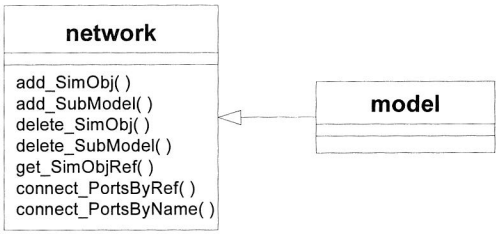


Abb. 64: Das Interface model

Da dies aber einem Klienten die Unterscheidung dieser beiden Klassen unmöglich machen würde, wird für die Modellobjekte ein Interface erstellt, das von *network* abgeleitet ist und den Namen *model* trägt (Abb. 64). Es verfügt weder über Attribute noch über Operationen und dient lediglich zur Unterscheidung der Modellobjekte von den Netzwerkobjekten.

6.4 Interface für das Kapseln der Simulationsdienste

Das Kapseln der Simulation geschieht mit dem Interface *simserver* (Abb. 65). Dieser bietet die grundlegenden Funktionen der Simulation (Starten, Anhalten, Wiederaufnehmen, Beenden) innerhalb der Architektur an. Der *simserver* ermöglicht auch die Abbildung einer Organisationsstruktur, also eines Modellobjekts, in den gekapselten Simulator.

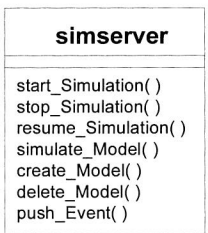


Abb. 65: Das Interface simserver

Über das *simserver*-Interface erfolgt auch die Weiterleitung von Ereignismeldungen an den gekapselten Simulator. Das Interface bietet dazu die Operation *push_Event* an. Indem die *Proxys* (5.2.1) diese Operation benutzen, können sie die von ihnen empfangenen Ereignismeldungen an die Simulation weiterleiten.

Neben der Steuerung eines Simulationssystems kann *simserver* zusätzlich als Erweiterung eines Wrapper-Objekts angesehen werden, da durch die Simulation bereitgestellte Daten ebenfalls als Attributobjekte bereitgestellt werden können.

6.5 Interfaces für die Implementierung der Objektfabriken

Um die Objekte der Architektur zur Laufzeit erzeugen zu können, wurden spezielle Interfaces realisiert, mit denen die Erzeugung der von einem Klienten benötigten Objekte möglich ist. Die Klassen, deren Objekte auf Wunsch eines Klienten erzeugt werden, sind Modell-, Kommunikations- und Proxyobjekte. Die erstellten Interfaces, folgen dem Entwurfsmuster der *Objektfabrik* [104] .

6.5.1 Das Interface proxyFactory

Für die Erzeugung der *Proxys* wurde das Interface *proxyFactory* erstellt (Abb. 66). Es ermöglicht seinen Klienten mit entsprechenden Operationen die zielgerichtete Erzeugung von *Proxy*-Objekten. Die *Objektfabriken*, die das *proxyFactory*-Interface implementieren, werden im weiteren Verlauf dieses Kapitels auch Proxyfabriken genannt.

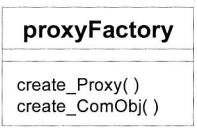


Abb. 66: Das Interface proxyFactory

Mit der ersten Operation des *proxyFactory* kann ein *Proxy* für ein existierendes Kommunikationsobjekt erstellt werden. Die Operation erzeugt einen *Proxy*, registriert ihn im Kommunikationsobjekt und gibt die Referenz des erstellten *Proxys* an den Klienten weiter. Die zweite Operation erzeugt mit dem Interface *modelFactory* (6.5.2) ein neues Kommunikationsobjekt. Daraufhin wird ein *Proxy* erzeugt und im Kommunikationsobjekt registriert. Der Klient erhält abschließend die Referenz des erzeugten und registrierten *Proxys*.

6.5.2 Das Interface modelFactory

Das für die Erzeugung der Modelle und der Kommunikationsobjekte erstellte Interface trägt den Namen *modelFactory* (Abb. 67). Es ermöglicht seinen Klienten die Er-

zeugung der obengenannten Objekte sowie den Zugriff auf Informationen über diese. Die *Objektfabriken,* die dieses Interface implementieren, werden im weiteren Verlauf dieses Kapitels als Modellfabriken bezeichnet. Sie ermöglichen, zur Laufzeit Modelle zu erzeugen, zu vernichten und zu verwalten. Die Verwaltung bezieht sich darauf, Informationen der *modelFactory* von einem Klienten aus zu erreichen und einzelne Modellobjekte gezielt ansprechen zu können.

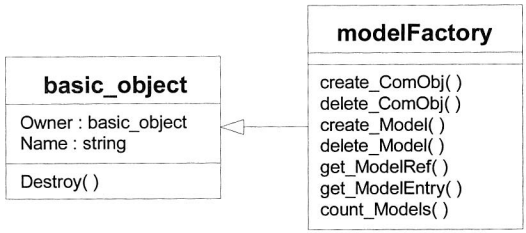


Abb. 67: Das Interface modelFactory

6.6 Interaktion zwischen den Objekten der Architektur

Nachdem nun für alle Objekte der Architektur zur Kopplung von Simulation und Datenquellen passende Interfaces definiert wurden (Abb. 68 gibt eine Gesamtübersicht), findet die Betrachtung des über diese Interfaces erfolgenden Nachrichtenaustausches statt.

Als erstes wird beschrieben, wie die Simulation als der Klient der Architektur die zum Datenaustausch benötigten Proxyobjekte erzeugen kann. Danach erfolgt eine Ablaufbeschreibung, mit denen die von der Simulation benötigten Daten aus der Datenschicht akquiriert werden. Anschließend wird der simulationsseitige Zugriff auf die vorher akquirierten Daten beschrieben.

Zuletzt erfolgt die Betrachtung der Vorgänge, mit denen Informationen über Störungen (asynchrone Ereignisse) der realen Fertigungskomponenten durch die Architektur an das Simulationswerkzeug geleitet werden.

6.6.1 Erzeugen eines Proxys

Die von der Simulation benötigten *Proxys* können auf zwei unterschiedliche Arten von der Proxyfabrik erzeugt werden. Bei der ersten wird ein *Proxy* zusammen mit einem Kommunikationsobjekt erzeugt. Dies ermöglicht es, aus der Simulation heraus eine flexible, zielgerichtete Verbindung mit den Systemen der Datenschicht aufzubauen. Dieser Fall wird in Abb. 69 veranschaulicht.

Bei der zweiten Art hingegen wird von der Proxyfabrik für ein vorhandenes Kommunikationsobjekt ein entsprechender *Proxy* erstellt und beim Kommunikationsobjekt registriert. Die hier stattfindenden Vorgänge werden durch die Beschreibung des ersten Falls abgedeckt. Er unterscheidet sich dadurch, dass bei (1) der Aufruf der Operation *create_Proxy* erfolgt und die Schritte (2) und (3) übersprungen werden.

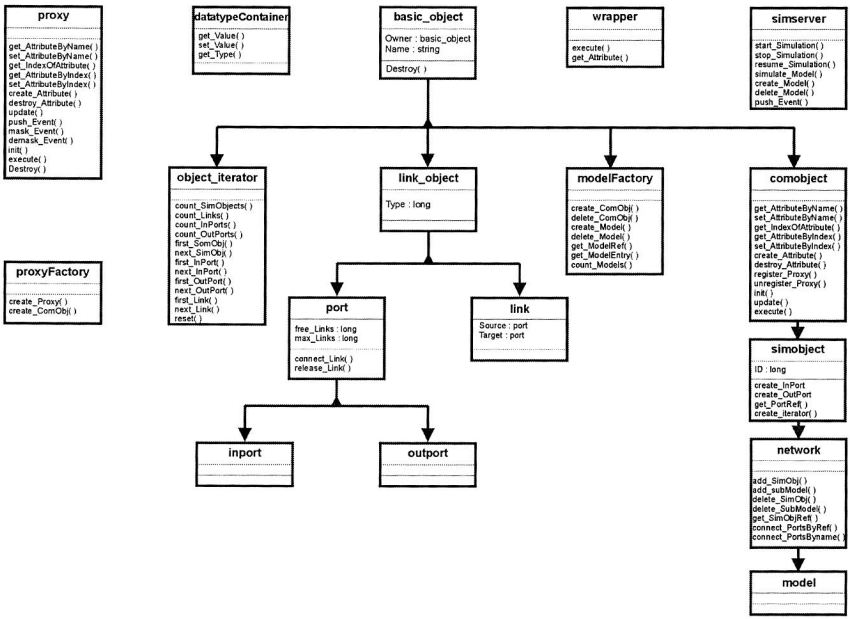


Abb. 68: Gesamtübersicht der realisierten Interfaces und Klassen

Die Simulation initiiert die Erzeugung des *Proxys* durch das Aufrufen der *create_ComObj*-Operation des *proxyFactory*-Interfaces (1). In der Proxyfabrik erfolgt nun ein Aufruf der gleichnamigen Operation des *modelFactory*-Interfaces (2). Dies hat zur Folge, dass die Modellfabrik ein Kommunikationsobjekt erzeugt (3) und die Referenz dieses Objekts an die Proxyfabrik zurückgibt. Diese erzeugt nun den *Proxy* (4) und registriert ihn beim Kommunikationsobjekt, dessen Referenz sie von der Modellfabrik erhalten hat (5). Während der Registrierung wird der Inhalt aller Attributobjekte des originären Kommunikationsobjekts in die entsprechenden *Proxy*-Attributobjekte übertragen (6), indem in *register_Proxy* für jedes Attributobjekt des *Proxys* ein Aufruf der *update*-Operation des *proxy*-Interfaces erfolgt, der das entsprechende Attributobjekt aktualisiert. Danach wird die Referenz des erzeugten und registrierten *Proxys* an die Simulation zurückgeben.

Auch nach der Registrierung aktualisiert das originäre Kommunikationsobjekt nun die entsprechenden Attributobjekte seiner *Proxys* mittels der *update*-Operation des *proxy*-Interfaces, sobald sich eines seiner Attributobjekte geändert hat. Ebenso wird von den *Proxys* bei der Änderung eines ihrer Attributobjekte das originäre Objekt durch die *update*-Operation des *comobject*-Interfaces aktualisiert. Damit ist sichergestellt, dass die Datenbestände der beteiligten Objekte konsistent bleiben.

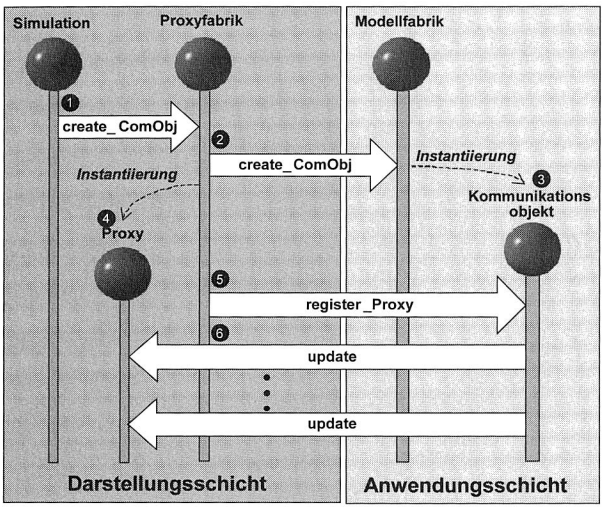


Abb. 69: Erzeugung eines Proxys

6.6.2 Initialisierung

Die Initialisierung eines Kommunikationsobjekts mit den Informationen der Datenschicht (Abb. 70) erfolgt, indem die simulationsrelevanten Daten eines Datenschicht-Systems (Datenbank oder Maschine) in die Attributobjekte des Kommunikationsobjekts und aller in ihm registrierten *Proxys* übertragen werden.

Zudem kann während der Initialisierung auch eine Anbindung an den *Event Service* erfolgen. Durch ihn werden die Ereignismeldungen geleitet, die bei der Störung einer realen Fertigungskomponente generiert werden.

Die Initialisierung wird gestartet, indem aus der Simulation die *init*-Operation des *proxy*-Interfaces aufgerufen wird (1). Dabei muss mit dem *policy*-Argument dieser Operation festgelegt werden, welche Strategie bei der Initialisierung angewendet werden soll. Die möglichen Belegungen dieses Arguments wurden in Tabelle 4 aufgeführt.

Im weiteren Verlauf dieser Ausführungen wird davon ausgegangen, dass das *policy*-Argument festlegt, dass als erstes die Anbindung an eine Datenbank erfolgen soll

(Zugriffsmethode 4). Falls dies nicht möglich ist, wird ein Anbindungsversuch an die reale Fertigungskomponente vorgenommen, die ebenfalls über die benötigten Daten verfügt. In diesem Fall wird außerdem versucht, eine Anbindung an den *Event Service* zu erreichen.

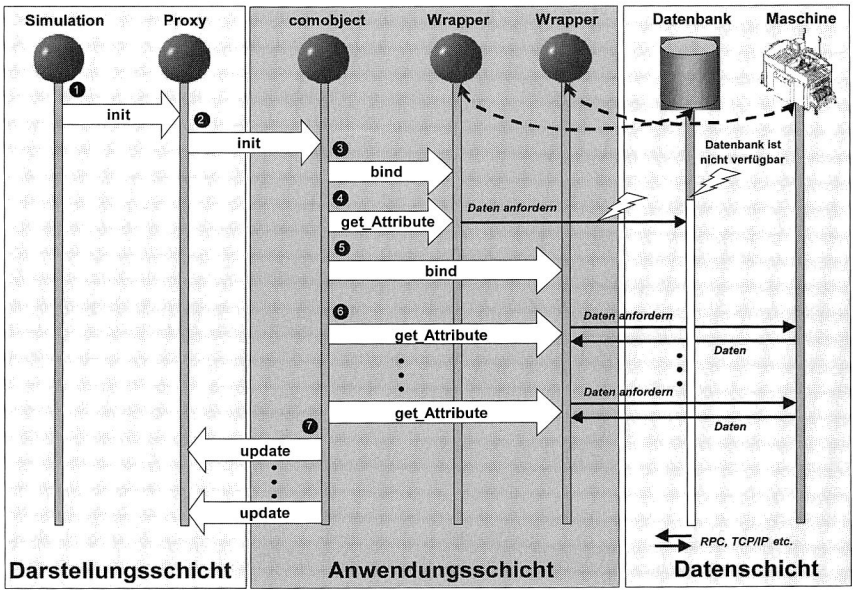


Abb. 70: Die Initialisierung eines Kommunikationsobjekts

Der *init*-Aufruf wird vom *Proxy* an sein Kommunikationsobjekt delegiert (2). Hier erfolgt die eigentlich Verbindung mit der Datenschicht. Durch den Aufruf von *bind* (3) wird versucht, eine Verbindung mit dem Wrapper der Datenbank zu bekommen. Nach der erfolgreichen Bindung an den Wrapper versucht das Kommunikationsobjekt mit der *get_Attribute*-Operation (4) des wrapper-Interfaces die verlangten Daten aus der Datenbank zu beschaffen. Falls an dieser Stelle ein Fehler auftritt, z.B. wenn die vom Wrapper gekapselte Datenbank zu diesem Zeitpunkt nicht verfügbar ist, erfolgt ein Bindungsversuch (*_bind*) (5) an den Wrapper, der die reale Fertigungskomponente kapselt. War die Anbindung an die Maschine erfolgreich, erfolgt bedarfsweise die Verbindung mit dem *Event Service*. Danach wird der wrapper aufgefordert, die verlangten Informationen aus der Maschine zu übertragen. Dies erfolgt sooft, bis alle simulationsrelevanten Daten in den Attributobjekten des Kommunikationsobjekts hinterlegt wurden.

Nachdem alle Daten aus der Datenschicht akquiriert wurden, werden die Attributobjekte aller registrierten *Proxys* durch die *update*-Operation (7) des *proxy*-Interfaces aktualisiert. Nach der Aktualisierung der *Proxys* ist die Initialisierung beendet und die Simulation kann auf die beschafften Daten zurückgreifen, indem sie mit den entsprechenden Operationen des *proxy*-Interfaces den Inhalt der Attributobjekte des *Proxys* anfordert. Eventuelle Störungen der Fertigungskomponente werden vom Wrapper an das Kommunikationsobjekt und dann über den *Proxy* an die Simulation weitergeleitet.

6.6.3 Zugriff auf die Attributobjekte

Der simulationsseitige Zugriff auf die Informationen eines Kommunikationsobjekts erfolgt durch die in ihm registrierten *Proxys*. Die dabei stattfindenden Interaktionen sind unterschiedlich und hängen davon ab, ob der Zugriff lesend oder schreibend erfolgt. In Abb. 71 wird der schreibende Zugriff dargestellt, in Abb. 72 der lesende.

Schreibender Zugriff: Um die Informationen eines *Proxys* zu verändern, benutzt die Simulation die Operation *set_AttributeByName* (1). Nachdem sie aufgerufen wurde, schreibt der *Proxy* mit der Operation *set_Value* den neuen Wert in das entsprechende Attributobjekt (2).

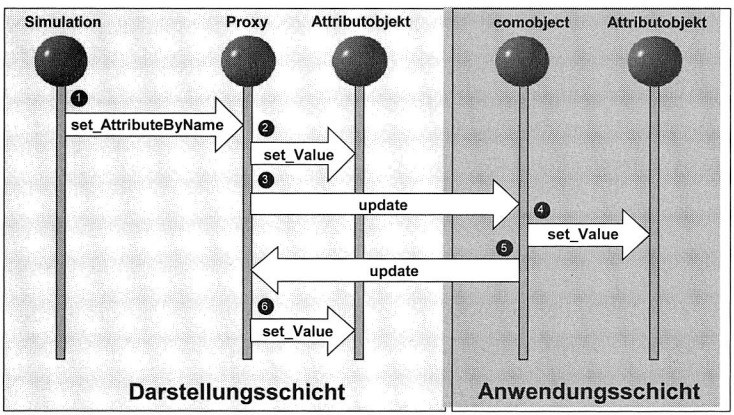


Abb. 71: Schreibender Zugriff auf die Daten eines Kommunikationsobjekts

Um die Veränderung des *Proxys* dem originären Kommunikationsobjekt mitzuteilen, benutzt der *Proxy* die *update*-Operation des *comobject*-Interfaces (3). Durch Ausführen der Operation *set_Value* aktualisiert das Kommunikationsobjekt daraufhin den Wert des entsprechenden Attributobjekts und initiiert mit der *update*-Operation (5) des *proxy*-Interfaces die Aktualisierung aller in ihm registrierten *Proxys*. In *update*

startet die Operation *set_Value* (6) und aktualisiert damit das entsprechende Attributobjekt auf den vom Kommunikationsobjekt erhaltenen Wert.

Lesender Zugriff: Da der *Proxy* beim originären Kommunikationsobjekt registriert ist, kann davon ausgegangen werden, dass die Daten des *Proxys* mit den Daten des Kommunikationsobjekts übereinstimmen. Der simulationsseitige Zugriff erfolgt durch die Operation *get_AttributeByName* (1). Sie holt durch *get_Value* (2) die aktuellen Daten des entsprechenden Attributobjekts und gibt dann diese der Simulation als Ergebniswert zurück.

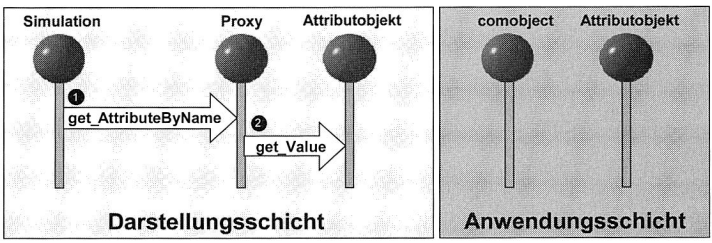


Abb. 72: Lesender Zugriff auf die Informationen eines Kommunikationsobjekts

6.6.4 Steuerung eines Datenschicht-Systems

Die Operation *execute* des *proxy*-Interfaces ermöglicht der Simulation die Steuerung der Systeme der Datenschicht. In Abb. 73 wird dies dargestellt. Um dies zu erreichen, wird von der Simulation die *execute*-Operation des *Proxys* aufgerufen (1), der mit dem System verbunden ist, das gesteuert werden soll. Im *Proxy* erfolgt nun die Delegation von *execute* auf die gleichnamige Operation des originären Kommunikationsobjekts (2). Daraufhin ruft das originäre Kommunikationsobjekt die *execute*-Operation des Wrappers auf und löst damit das von der Simulation verlangte Verhalten aus (3).

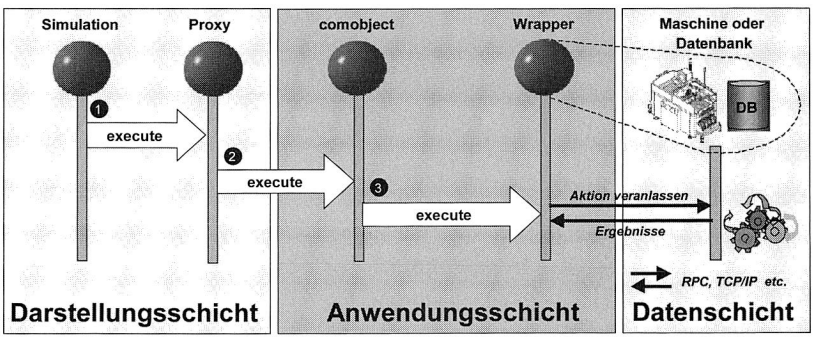


Abb. 73: Steuerung eines Datenschicht-Systems durch die Simulation

Eventuelle Ergebnisse des Datenschicht-Systems werden über die Rückgabewerte der *execute*-Aufrufe bis an die Simulation als Attributobjekt zurückgegeben.

6.6.5 Übertragung einer Ereignismeldung

In Abb. 74 ist ein Interaktionsdiagramm abgebildet, das die Kommunikation bei der Übertragung einer Ereignismeldung von der realen Fertigungskomponente in die Simulation mit dem *Event Service* veranschaulicht: Nach dem Eintreten eines Ereignisses, z.B. der Störung einer Maschine (1), gelangen Informationen, die das Ereignis beschreiben, in den Wrapper, der die Maschine kapselt. Der Weg, auf dem dies erfolgt, ist abhängig von der Maschine und den von ihr angebotenen Schnittstellen. Der Wrapper erzeugt nun aus den erhaltenen Störinformationen eine Ereignismeldung und sendet diese über den *Event Service* an alle mit ihm verbundenen Kommunikationsobjekte (2).

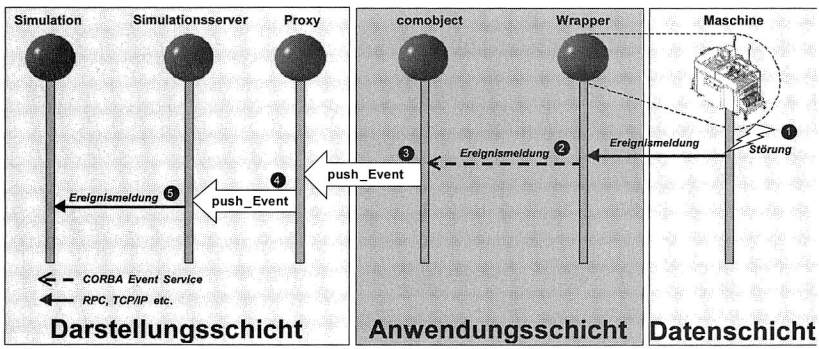


Abb. 74: Übertragung einer Ereignismeldung an die Simulation

Die Kommunikationsobjekte, die die Ereignismeldung des Wrappers empfangen, übertragen diese an alle in ihnen registrierten *Proxys*. Dies erfolgt, indem sie die *push_Event* Operation des *proxy*-Interfaces benutzen (3). Die *Proxys* senden die empfangene Ereignismeldung an den Simulationsserver, indem sie die *push_Event*-Operation des *simserver*-Interfaces aufrufen (4). Daraufhin überträgt der Server die von den *Proxys* empfangenen Meldungen an den in ihm gekapselten Simulator (5). In der Simulation können nun die Informationen der empfangenen Ereignismeldung genutzt werden, um in einem Simulationslauf die Folgen der Störung abzuschätzen.

6.7 Realisierung der Architektur

Die konzipierte Architektur war zur Bewertung zu realisieren. Die gewählte Vorgehensweise hat vorgesehen, zunächst die Interfaces der Kap. 6.1-6.6 in die IDL (*interface definition language*) von CORBA zu übertragen.

Diese Beschreibungen wurden anschließend durch Compiler in C++ bzw. JAVA übertragen (Abb. 75). Diese bildeten dann die Implementierungsgrundlage für die Funktionalitäten der Architektur.

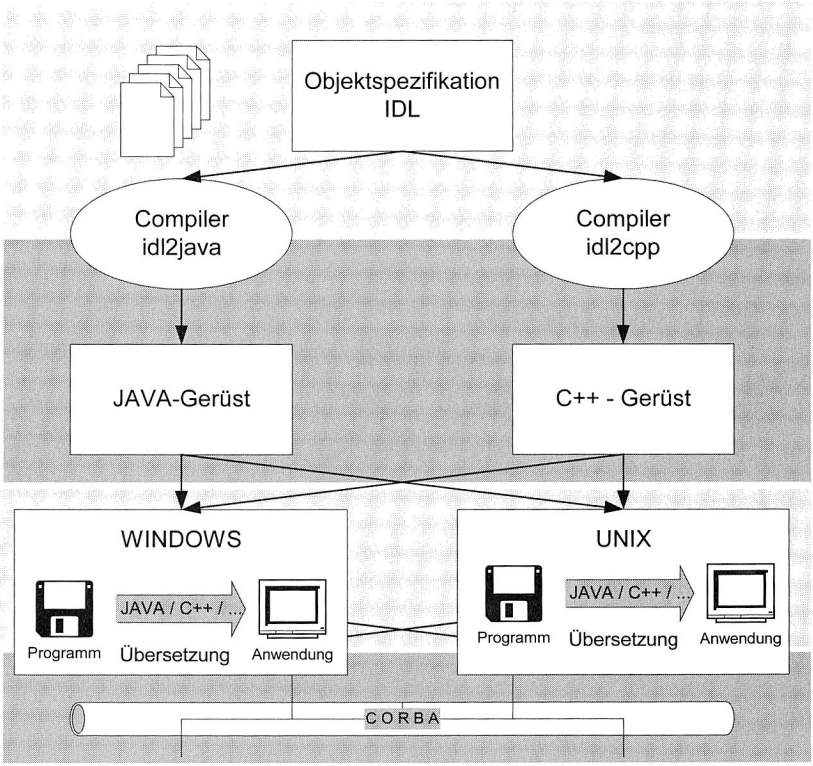


Abb. 75: Programmier- und Plattformunabhängigkeit der Objektspezifikation durch IDL-Compiler

Die Wahl von CORBA und IDL ermöglichte dabei die geforderte Plattform- und Sprachunabhängigkeit, weil das IDL-*language-mapping* ohne Rückgriff auf eine konkrete Zielplattform verwendet werden kann. Zusätzlich kann besonders der Einsatz von JAVA die Austauschbarkeit von Softwarekomponenten aufgrund des Konzepts der virtuellen Maschine schnell und einfach vorantreiben. Die Programmiersprache C++ findet in dieser Architektur demzufolge nur dann Verwendung, wenn die Ausführungsgeschwindigkeit eines Teilmoduls wichtig ist. Die vom IDL-Compiler erzeugten Programmteile gewährleisten eine programmiersprachenübergreifende Schnittstellenkompatibilität, d.h. innerhalb der Architektur kann beispielsweise ein C++-Methodenaufruf von einer JAVA-Methode korrekt ausgeführt werden und umgekehrt.

6.7.1 Anwendungsschicht

Die Module der Anwendungsschicht stellen den Kern des Systems dar. Diese umfassen die Verwaltung der Kommunikations-, Simulations-, Modell-, und Strukturobjekte. Dies bedeutet weiterhin, dass abgesehen von der Administration und der Parametrierung beim Start der Anwendungsschicht keine Benutzereingriffe notwendig oder möglich sind.

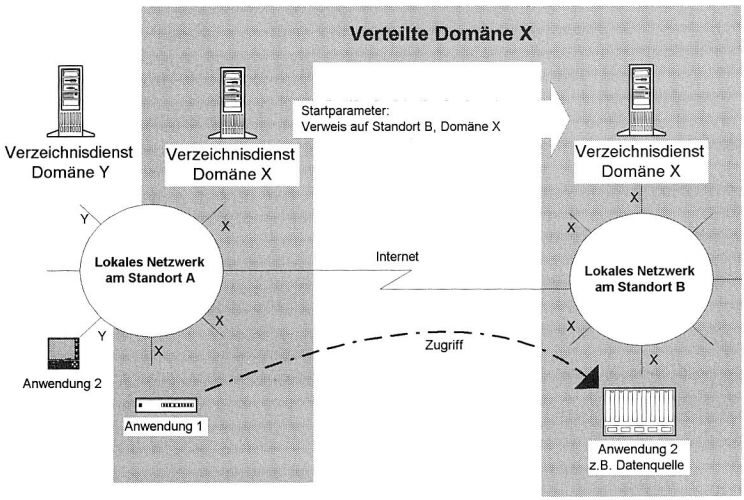


Abb. 76: Zwei Verzeichnisdienste auf verschiedenen, über das Internet verbundenen Netzwerken mit gemeinsamem Namensraum

Die Startparameter legen die Namensräume der Modellfabriken fest (z.B. "Factory-Schottky") und ggf. die IP-Adresse und die Port-Adresse eines weiteren Anwendungsschicht-Systems.

Das System ist so eingerichtet, dass es einen dynamischen, verteilten Verzeichnisdienst verwendet, der u.U. parallele Domänen (z.B. gleichzeitiger Betrieb eines Test- und eines Produktivsystems) an unterschiedlichen Standorten oder Netzwerksegmenten anbietet. Dieser Verzeichnisdienst erlaubt anschließend, beliebige Objekte innerhalb der Architektur zu lokalisieren und zu nutzen. Abb. 76 verdeutlicht dies.

Nachdem die Anwendungsschicht gestartet wurde, sind die in Kap. 6.3 spezifizierten Funktionen innerhalb des gesamten Frameworks verfügbar, d.h. die Simulation kann Kontakt zu Datenquellen aufnehmen oder der Benutzer kann in den Datenbeständen navigieren, Modelle im scratch-board erstellen und im Simulationssystem aufbauen lassen. Beispielhaft sei dies in Abb. 77 verdeutlich: Im Fenster links ist der Start der Anwendungsschicht mit Parameterübergabe und Kontrollmeldungen zu erkennen

(Plattform: Windows NT). Im zweiten Fenster wurde in Testprogramm (Windows 2000) gestartet, welches die grundlegenden Funktionen beim Aufbau eines Modells im scratch-board nachbildet. Erwähnenswert ist hier besonders die Nennung eines zweiten Domänenstandorts (HP/UX, anders Subnetz), da nach dem Start und der Bildung der aufgeführten Struktur ein solches Modell an beiden Standorten verfügbar ist. Dies ist im dritten Fenster zu erkennen, welches das Ergebnis der Ausgabe eines weiteren Testprogramms in der erwähnten remote-domain ist (HP/UX). Zuletzt wird im vierten Fenster der erste Prototyp des Navigators (ModelBrowser) gezeigt, mit dem die vorher erstellen Strukturen visualisiert werden können.

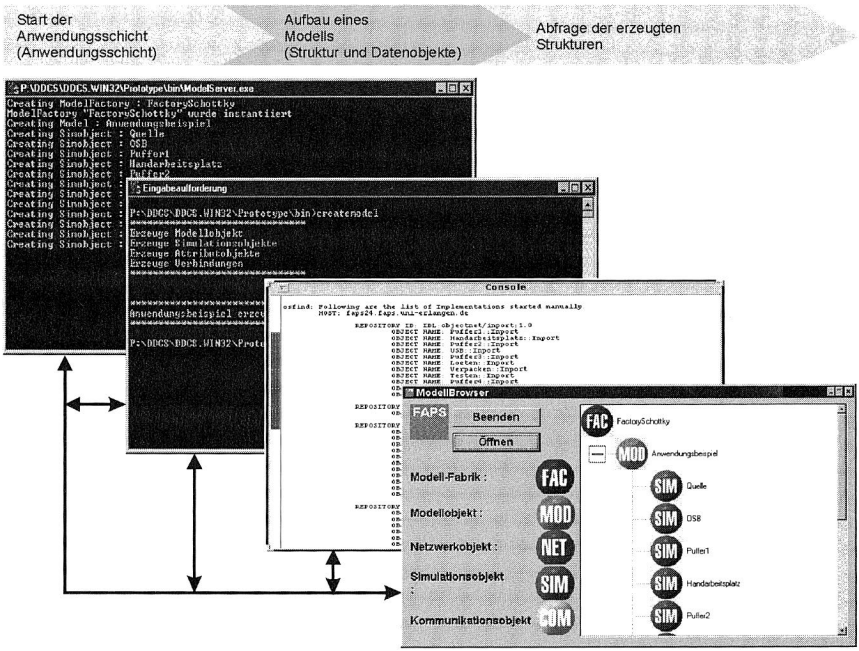


Abb. 77: Exemplarische Aufrufkette zwischen Anwendungsserver und weiteren Prozessen auf unterschiedlichen Betriebssystemen

Die Darstellung macht ebenfalls deutlich, dass die Anwendungsschicht in einem einfachen Fenster abläuft - Benutzereingaben sind an dieser Stelle zur Laufzeit nicht erforderlich und die notwendigen Programme können ggf. als Dienste im Hintergrund laufen.

Die Nutzung der Anwendungsschicht erschließt sich besonders über die Anwender- und Administrationswerkzeuge, sowie über das Navigationsmodul, welches in Kap. 6.7.4 genauer vorgestellt wird.

6.7.2 Darstellungsschicht und Simulationssystem

Die Umsetzung des vorgestellten Konzepts auf Seiten des Simulationssystems (Darstellungsschicht) erfolgt in fünf Schritten. Diese sind so aufeinander abgestimmt, dass eine Erweiterung der meisten gängigen Simulationswerkzeuge möglich ist, sofern diese über einen ein- und einen ausgehenden Datenaustauschkanal oder über eine systemerweiternde Schnittstelle, z.B. DLL oder zur Laufzeit einbindbare *sharded libraries*, verfügen.

Eine Auswertung einer aktuellen Marktstudie zeigt, dass dies bei den marktgängigen Produkten gegeben ist [141]. Die Realisierung des vorliegenden Konzepts erfolgte mit dem Simulationssystem Simple++ (eM-Plant) auf den Plattformen Windows NT und HP-UX aufgrund der derzeitigen Marktführerschaft dieses Produkts in Deutschland.

Schritt 1 - Erstellen eines Kopplungsmoduls für das Simulationssystem:

Zunächst wird das Kopplungsmodul erstellt, mit dem die Bedienbarkeit des Simulationssystems von außen möglich wird. Die Funktionalitäten umfassen 'Start', 'Stop', 'Anhalten' und 'Fortsetzen' eines Simulationslaufs, sowie das 'Erstellen von Modellelementen' innerhalb des Systems. Das Ergebnis dieses Schritts ist die Komponente "Simulations-Server".

Schritt 2 - Erweiterung des Bausteinvorrates des Simulationssystems:

Voraussetzung für die Erstellung und spätere Nutzung der Modellelemente ist eine Erweiterung des Bausteinumfangs des Simulators. Diese Bausteine haben die Aufgabe, als Repräsentanten der realen Datenobjekte sowohl funktional als auch strukturell zu fungieren.

Die Erweiterung muss dabei so erfolgen, dass bestehende Bausteine substituiert werden, d.h. die ursprüngliche Funktionalität bleibt bestehen und wird für die beschriebenen Frameworkzwecke erweitert.

Zu nennen ist hier einerseits die Referenz eines Datenquellenobjekts, sowie Verwaltungsmodule und Methoden, mit denen der Datenaustausch und die externe Steuerung dieses Bausteins, beispielsweise die Signalisierung einer Störung, erfolgt.

Schritt 3 - Erweiterung des Befehlsvorrates des Simulationssystems:

Simulatorintern muss eine Befehlserweiterung vorgenommen werden, die die zusätzlichen Funktionen der Architektur durch Aufrufe erschließt. Hierzu gehören alle Aufgaben der Initialisierung, Konfiguration und Steuerung des Austauschs zwischen dem Simulationssystem (Darstellungsschicht) und der Anwendungsschicht (als Repräsentant der Datenschicht). Die meisten der realisierten Funktionen müssen nicht

vom Anwender aufgerufen werden. Diese sind vielmehr in den erweiterten Bausteinen aus Schritt 2 gekapselt.

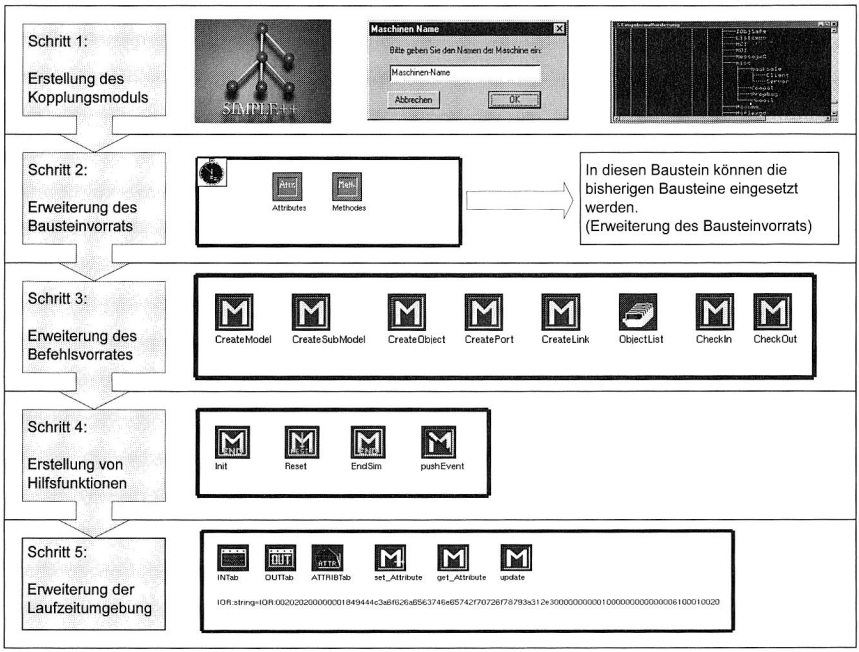


Abb. 78: Schrittweise Erweiterung der Darstellungsschicht (Simulationssystem)

Schritt 4 - Erstellen von Hilfsfunktionen für das Simulationssystem:

Der Aufbau von Simulationsmodellen aus der Anwendungsschicht heraus verlangt eine Kapselung der einzelnen Abläufe im Simulationssystem. Entsprechend sind Hilfsfunktionen zu erstellen, die aus Architektursicht als Programmierschnittstelle die Entitäten Port, Link, Modell, Submodell und Objekt in Objekte des Simulators umsetzen. Demzufolge handelt es sich hierbei um eine Bibliothek, die die Modellbildung simulatorunabhängig und abstrahiert beschreibt. Die Aufrufe der Methoden erfolgen dabei über das Kopplungsmodul, initiiert durch den Modellgenerator.

Schritt 5 - Erweiterung der Laufzeitumgebung des Simulationssystems:

Die Nutzung der Datenquellen zur Simulationslaufzeit, d.h. Datenabfrage und asynchrone Ereignismeldungen, erfolgt durch einen einfachen Aufruf innerhalb des entsprechenden Modellelements (siehe Schritt 4). Durch die Befehlserweiterung aus Schritt 3 gelangt dieser Aufruf an ein weiterhin zu erstellendes Modul, welches dann die eigentliche Datenbeschaffung auslöst.

Dieses Modul fungiert dabei als lokaler Proxy des Simulationssystems zur Anwendungsschicht, der darüber hinaus auch für die erforderliche Datentypenkonvertierung und die koordinierte Kommunikation zuständig ist. Ein aktiver Eingriff des Simulationssystems ist nicht notwendig, was maßgeblich zur Unabhängigkeit von einem speziellen Simulationssystem beiträgt.

Durch die Befehlserweiterung kann dabei zunächst aus einem Simulationsbaustein manuell oder automatisch (durch den Generator) im Proxy ein Katalog von 'interessanten' Daten definiert werden, der anschließend die Grundlage für eine Update-Operation ist, die das notwendige Datenmaterial aus den Kommunikationsobjekten beschafft. Eine Übersicht der beschriebenen fünf Schritte ist aus Abb. 78 zu entnehmen.

6.7.3 Datenschicht - Prototypen der *Wrapper*

Die Objekte der Datenschicht werden in der Architektur durch Wrapper zugänglich gemacht. Diese Module haben die Aufgabe, als Vermittler und Umsetzer zwischen den Formaten der Datenquelle und der Architektur zu operieren. Architekturseitig ist eine Datenquelle ausschließlich durch einen zugeordneten Wrapper sichtbar.

Weil die Menge der potentiellen Datenquellen äußerst heterogen ist, kann an dieser Stelle keine konkrete Implementierung eines Wrappers beschrieben werden. Aus dem gleichen Grund ist der Funktionsumfang eines Wrappers zur Nutzung der Architektur auf nur zwei Methoden beschränkt: lesen von Attributen und Ausführen von Operationen der gekapselten Datenquelle. Die Flexibilität dieses Ansatzes wird erreicht, indem die Wrapper-interne Interpretation der auszuführenden Befehle entsprechende Objektattribute erzeugt, die wiederum durch einen Klienten lesbar sind. Die Interpretation erfolgt objektspezifisch. Realisierungen wurden beispielhaft für einige wichtige Funktionen der GEM-Spezifikation auf einem Linienrechner einer Bestückmaschine (Siemens SIPLACE S23) vorgenommen. Sie erfolgen immer auf der Basis folgender IDL-Interfacebeschreibung:

```
module objectnet {
      exception InvalidParameter {};
      exception NotFound {};
      exception NotPossible {};

interface wrapper{
      any execute(in string command, in any data)
      raises(NotFound,InvalidParameter,NotPossible);
      any get_Attribute( in string name)
      raises(NotFound,InvalidParameter,NotPossible);
      };

};
```

Weitere Implementierungen erfolgten für ODBC, SQL, TCP/IP und DDE. Ebenso wurde ein Grundgerüst in den Programmiersprachen JAVA und C++ erstellt, welches die Umsetzung weiterer Wrapper-Objekte stark vereinfacht. Hierdurch war es auch möglich, durch den Einsatz einer CORBA/COM-Bridge auf Datenquellen aus OPC-Servern zuzugreifen, womit zum gegenwärtigen Stand der Technik ein äußerst breites Spektrum potentieller Datenquellen genutzt werden kann.

Abb. 79 stellt dies am Beispiel eines Datenbank-Wrappers dar, von dem einige Daten abgefragt wurden.

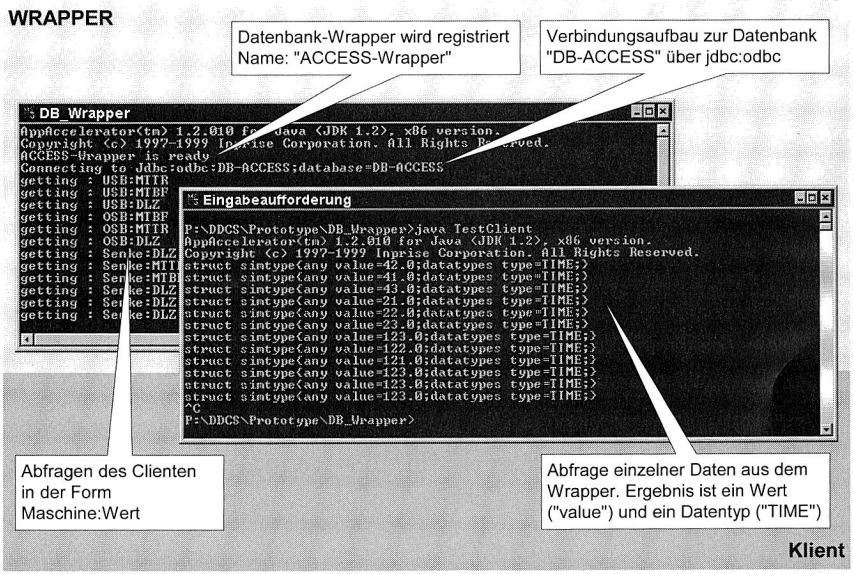


Abb. 79: Einsatzbeispiel eines Datenbank-Wrappers (oben) und Abfrage einiger Daten durch einen Test-Client (unten)

In der Architektur ist es weiterhin vorgesehen, dass ein Kommunikationsobjekt nicht direkt mit einem Wrapper eines realen Datenobjekts verbunden sein muss. Vielmehr sind auch Konfigurationen realisierbar, bei denen die bereitgestellten Daten verschiedener Wrapper zunächst von einen weiteren Wrapper vorverarbeitet (verdichtet/konvertiert) werden. Damit wird es beispielsweise möglich, dass mehrere Wrapper zunächst zeitpunktbehaftete Informationen (Sensorsignale: Ankunft an Station A/B) erfassen und diese anschließend in zeitraumbehaftete Parameter (Durchlaufzeit zwischen A und B) verdichtet und bereitgestellt werden.

6.7.4 Navigation, Generator und der Dienst zur Datensicherung

Die Kontrolle der wichtigsten Funktionen der Kommunikationsarchitektur wurde für den Benutzer unter einer einheitlichen Oberfläche zusammengefasst. Dies betrifft die wichtigsten Steuerungen der Architektur (Starten, Beenden, Status und Abfrage aller aktiven Dienste).

Darüber hinaus finden sich hier die Navigationskomponente, das scratch-board und der Modellgenerator, mit denen der Anwender die datengetriebene Modellierung durchführt. Dazu wird zunächst eine hierarchische Sicht auf den Bestand verfügbarer Datenquellen dargestellt, welche nach den Kriterien Ressource, Prozess und Produkt ausgewählt werden kann. Anschließend können mit dem Werkzeug Teilbäume in das scratch-board übernommen und hier weiterbearbeitet werden. Zuletzt kann die hier erzeugte Struktur in die Anwendungsschicht und von dort weiter in das Simulationsmodell übertragen werden. Abb. 80 zeigt das Navigations- und Generatorwerkzeug, *NavGen* sowie das damit in Simple++ (eM-Plant) erzeugte Teilmodell.

Im Modul NavGen sind zusätzlich die Oberflächen für die Kontrolle des Dienstes zum Datenabgleich zwischen Datenquellen und einer Datenbank zu finden. Es wurde bereits beschrieben, dass bei der Unterbrechung der Kommunikation zu einer realen Datenquelle der Zugriff auf historisierte Daten vorzusehen ist, die statt der eigentlichen Daten verwendet werden. Dies geschieht zwar um den Preis einer geringeren Datenqualität (d.h. Aktualität), jedoch ist es für Simulationszwecke häufig besser, mit historisierten Daten zu arbeiten und gleichzeitig darauf hingewiesen zu werden, als über gar kein Datenmaterial zu verfügen.

Um die Erzeugung historischer Daten zu ermöglichen (Historisierung), wurden für die Architektur eigene Dienste entworfen und implementiert, die ausgewählte Daten aus Datenquellen in einer Datenbank sammeln:

- der erste Dienst sichert Daten periodisch, d.h. zu definierten Zeitpunkten werden die gesicherten Daten in einer Datenbank aktualisiert, bzw. ergänzt.
- der zweite Dienst arbeitet als Monitor, der Datenobjekte beobachtet und bei Änderungen diese in der Datenbank aktualisiert (Trigger)

Die Implementierung dieser Dienste erfolgt aus Benutzersicht als Erweiterung der Kommunikationsobjekte, die um die beschriebenen Überwachungs- und Datenverwaltungsfunktionalitäten erweitert wurden. Der Dienst selbst ist jedoch als Wrapper realisiert, der für die jeweilige Betriebsart angepasst werden muss.

Entsprechend kann der Benutzer im Modul NavGen die notwendige Parametrierung vornehmen. Dies verdeutlicht nicht nur die Arbeitsweise dieser Funktion, sondern

zeigt auch, wie die Kommunikation mit einem Wrapper hinsichtlich der Abfrage und Veränderung von Informationen erfolgen kann.

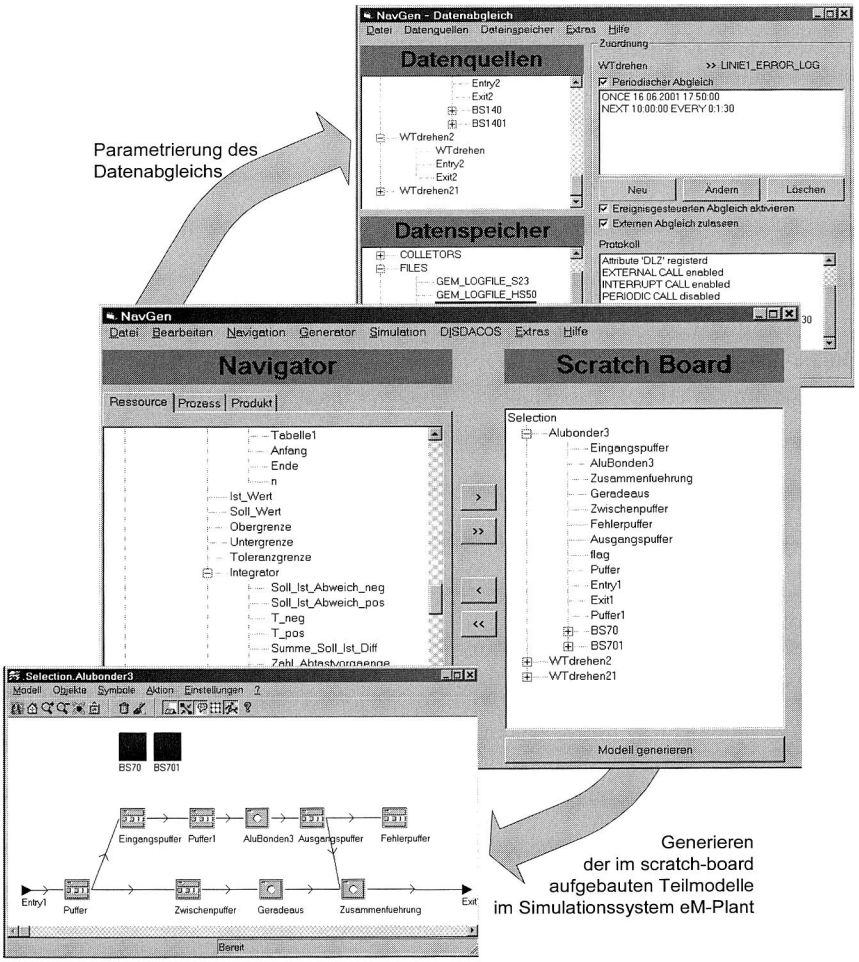


Abb. 80: Das Modul NavGen zur Navigation, Generierung von Simulations(teil)modellen und zur Parametrierung des Datenabgleichdienstes

6.8 Zusammenfassung

Dieses Kapitel hat die Konzeption und Realisierung der Funktionalität eines Kommunikationsframeworks zur Beschaffung simulationsrelevanter Daten vorgestellt. Der Aufbau von DISDACOS (Distributed Data Collection System) basiert auf einer drei-

schichtigen, verteilten und objektorientierten Modellierung, die für den Nutzer jedoch nicht sichtbar sein soll. Aus Anwendersicht wird das gewohnte Simulationswerkzeug lediglich erweitert und Informationen aus einem Produktions- oder Planungssystem sind 'on-demand' verfügbar. Hierfür reicht die Benennung einer Datenquelle(-ngruppe) aus. Hierbei sichern Mechanismen innerhalb der Architektur zu, dass der Datenakquisitionsprozess so robust wie möglich ausgeführt werden kann, da bedarfsweise auf alternative Datenquellen innerhalb einer Datengruppe umgeschaltet wird, falls eine spezifizierte Datenquelle nicht verfügbar sein sollte.

Die für eine solche umfassende Datenbereithaltung notwendige Organisation der Datenbestände erschließt durch die vorgestellten Administrations- und Navigationskomponenten zusätzlich die Möglichkeit einer datengetriebenen Modellierung, da durch den Einsatz umfassender Hierarchierungskonzepte problem- und zielbezogen die Grundzüge von Simulationsmodellen aus den vorhandenen Datenbeständen abgeleitet werden können. Gemeinsam mit einem integrierten Postprozessormodul (Modellgenerator) können diese Grundzüge anschließend ohne weiteren manuellen Aufwand im verwendeten Simulationssystem das entsprechende Simulationsmodell generieren. Dabei ist es unwesentlich, ob Bediener und Simulation am gleichen Standort operieren. Die Architektur sichert eine durchgängige Standort-, Plattform- und weitgehende Systemunabhängigkeit zu.

7 Überwachung von Betriebsdaten mit der Simulation

Die vorangegangenen Darstellungen haben den Kern eines informationslogistischen Architekturkonzepts beschrieben. Den Schwerpunkt haben die technischen Aspekte des integrierten Managements eines verteilten, heterogenen Systems von Datenquellen gebildet, die dem Simulationsexperten zur Verfügung gestellt werden. Die hier vorgestellten und realisierten Dienste sind auf die Kopplung zwischen Objekten im Planungsprozess oder eines Produktionssystems mit der Systemsimulation ausgerichtet. Das bedeutet, dass die bisherigen Ergebnisse als Sammlung von Hilfswerkzeugen die Basis für ein zusätzliches Anwendungsfeld der Simulation darstellen.

Um dieses System nun auf die Ausgangssituation (Kap. 2.4 und 4) anwenden zu können, reicht es jedoch nicht aus, einzig die Informationen aus dem Planung- und Betriebsprozess verfügbar zu machen. Vielmehr erschließen sich die Potentiale dieses Konzepts erst, wenn innerhalb der Simulation Methoden und Werkzeuge bereitgestellt werden, die die Nutzung der Architektur aus Anwendersicht ermöglichen. In den folgenden beiden Kapitel wird dazu die Laufzeitüberwachung von Kenngrößen im Simulationsmodell erarbeitet (Kap. 7.1) und anschließend der Einsatz aller Systemteile dieser Arbeit in einem umfassenden Szenario vorgestellt (Kap. 8), woraus sich dann ein online-Monitoring als praktische Anwendung einer informationslogistischen Architektur zur Akquisition simulationsrelevanter Daten ergibt (Abb. 81).

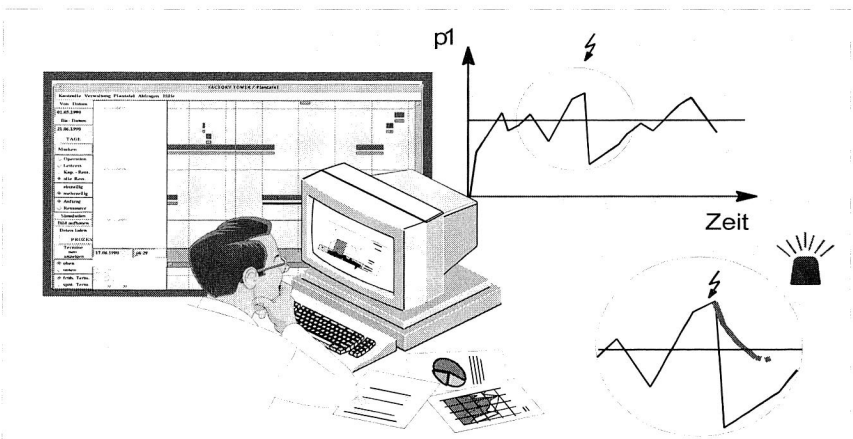


Abb. 81: Erfassung und -überwachung von Betriebsdaten

Die Überwachung von Betriebsdaten erfolgt in zwei Schritten. Zunächst findet eine Datenbeschaffung der Betriebsdaten durch DISDACOS statt, welche für eine perma-

nente Übertragung von Betriebsdaten in Parameter des Simulationsmodells sorgt. Anschließend werden diese Parameter innerhalb des Simulationssystems permanent auf die Einhaltung definierter Grenzen, bzw.- signifikanter Abweichungen, überwacht. Zentraler Arbeitspunkt dieses Kapitels ist es, dafür drei Referenzbausteine zu entwickeln, die unter dynamischen Gesichtpunkten die Parameterüberwachung ermöglichen.

7.1 Erkennung und Verarbeitung signifikanter Abweichungen von Prozessgrößen im Simulationsmodell

7.1.1 Definition "Störung"

Als Definition einer Störung soll die Abweichung zwischen Soll- und Ist-Größen herangezogen werden. Als Basis hierzu werden Kenngrößen genutzt, auf denen aufbauend relevante Abweichungen beschrieben werden sollen. Zu diesem Zweck wird ein Formalismus entwickelt, mit dem die Abbildung von erfassten Realdaten auf Kenngrößen mit dem Ziel der Störungserkennung erfolgen kann.

Die Beschreibung soll dabei gewährleisten, dass eine Abweichung nicht nur durch eine absolute Soll-/Istwertdifferenz charakterisiert wird, sondern dass die Verläufe einen zusätzlichen Beitrag zur Störungsidentifikation leisten. Dies ist sowohl für die Ursachenidentifikation als auch für eine mögliche Separation von Fehlern und Folgefehlern wichtig [142]. Weiterhin ist es nicht ausreichend, eine Störung nur zu erkennen, sondern es muss darüber hinaus auch möglich sein, die Störungsbeschreibung mit einer Verarbeitungsvorschrift (Regel) zu verbinden. Solche Regeln sind notwendig, um situationsbezogene Maßnahmen im Simulationsmodell anstoßen zu können. Dazu ist neben der Beschreibung der Störregeln ebenfalls zu klären, wie die erstellten Formalismen zum Zweck der Simulation eingesetzt werden, d.h. wie die technische Repräsentation und Auswertung in einem Simulationsmodell erfolgt. Das Ergebnis dieses Schritts ist konzeptionell ein Alarmmodul, welches Abweichungen tatsächlich erkennt und dann alle notwendigen Aktionen (Visualisierung, Diagnose, weitere Simulationsläufe) anstößt.

7.1.2 Systematik zur Erkennung von Störungen

Die Erkennung von Störungen zur Laufzeit eines Simulationsmodells oder in der Betriebsphase eines Produktionssystems ist nur dann sinnvoll, wenn deren Identifikation automatisch erfolgen kann. Demzufolge muss hierfür zunächst eine Systematik geschaffen werden. Diese gibt dann die Implementierungs- und Betriebsdetails für das notwendige Auswerte- (Alarm-)Modul vor. Bei der Beschreibung und Auswertung von Störungen ist im Zusammenhang mit dem Simulationseinsatz besonders zu be-

achten, dass die dynamischen Eigenschaften der betrachteten Systeme erhalten bleiben.

Diese Forderung wird in ähnlicher Form auf dem Gebiet der Regelungstechnik durch den PID-(Proportional, Integral, Differential) Regler abgedeckt [143]. Aufgabe solcher Regler ist es, die Regelabweichung der Regelstrecke (des Prozesses) von einer vorgegebenen Führungsgröße zu eliminieren. Das Einflussverhalten wird dabei durch Regelparameter bestimmt, die individuell abgestimmt werden müssen, d.h. die Stellgröße wird unter Berücksichtigung der erhaltenen Regelabweichung mit Hilfe der im Regler implementierten Regelalgorithmen bestimmt (Abb. 82).

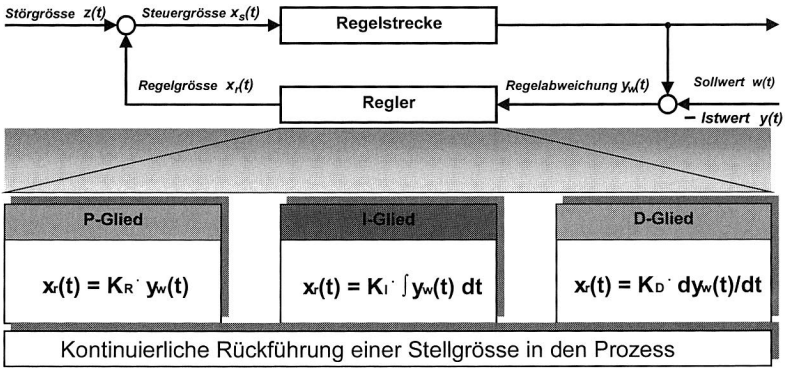


Abb. 82: Einsatz und Anwendung eines PID-Reglers

Der P-Anteil reagiert dabei auf einen zur Regelabweichung proportionalen Betrag, der I-Anteil erfasst die entstandene kumulierte Regelabweichung und der D-Anteil reagiert auf die Geschwindigkeit. Die einzelnen Anteile werden durch eine Linearkombination der PID-Komponenten zusammengefasst. Um das Konzept für zeitdiskrete Simulationssysteme nutzbar zu machen, sind zunächst die (analogen) mathematischen Operationen Differentation und Integration durch (digitale) Operation Differenzenquotient bzw. Summation zu ersetzen. Das führt letztlich zu einer Abtastregelung, bei der zu diskreten Zeitpunkten die Regelabweichung und entsprechende Stellgrößen bestimmt werden [143].

Die Idee der Störungsbeschreibung liegt nun darin, die einzelnen 'Regleranteile' als Kenngrößen anzusehen und diese zuletzt gegen vorgegebene - zulässige - Intervalle (Ober-/ Untergrenzen) zu testen. Das bedeutet, dass die resultierende Stellgröße ausschließlich bewertet und nicht in den Prozess zurückgeführt wird. Diese aus regelungstechnischer Sicht entstehende inline-Kopplung ist dadurch zu erklären, dass letztlich der Benutzer seine Maßnahme auswählt und diese Entscheidung dann erst

als Stellgröße in den Prozess (die Regelstrecke) einbringt, wie Abb. 83 verdeutlicht. Die vier Anteile (P, I, D und deren Linearkombination) sollen im folgenden beschrieben werden.

Im Proportionalanteil wird eine Soll-/Istwert-Abweichung ausschließlich skaliert, sodass die Signifikanz durch einen geeigneten Gewichtungsfaktor statisch voreingestellt werden kann. Der eigentliche Verlauf muss hierzu nicht berücksichtigt werden. Daher ist der Proportionalanteil so zu gestalten, dass aus Soll- und Istwert entweder die Differenz oder der Quotient ermittelt und dieses Ergebnis gewichtet wird. Die relevanten Informationen (Operation, Faktor, Datenquelle für Istwert, Referenz auf Sollwert) sind zusammen mit den Eckwerten des zulässigen Bereichs zu hinterlegen.

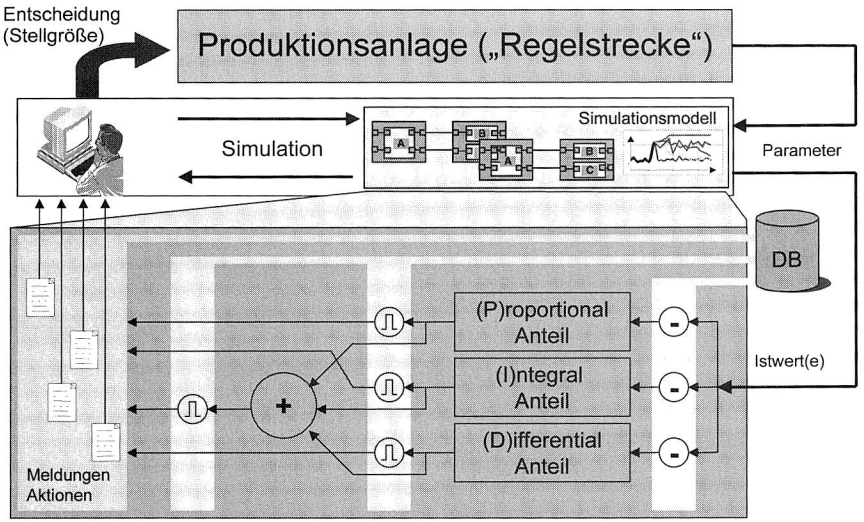


Abb. 83: `Struktur der Laufzeitüberwachung von Simulationsmodellen und Prozessen durch parametrierbare P, I und D - Komponenten

Die gleichen Informationen wie beim Proportionalanteil werden auch beim Differentialanteil verwendet werden. Diese Informationen müssen zusätzlich um den jeweiligen Betrachtungszeitraum, den Faktor Δt, erweitert werden.

Analog zum Proportionalanteil ist auch der Integralanteil zu gestalten. Zusätzlich muss hier berücksichtigt werden, dass der Integrationszeitraum eingeschränkt werden kann (begrenzte Historie).

Zuletzt wird vorgesehen, die einzelnen Anteile zur Störungserkennung in einer Linearkombination zusammenfassen zu können. Ein Gewichtungsfaktor ist an dieser Stelle nicht mehr nötig, da dieser bereits durch die einzelnen Faktoren einfließt, so-

dass letztlich nur noch die relevante Information *Referenz auf Sollwert* zusammen mit den Eckwerten des zulässigen Bereichs zu hinterlegen ist. In *Tabelle 5* wird das Ergebnis dieser Strukturierung dargestellt.

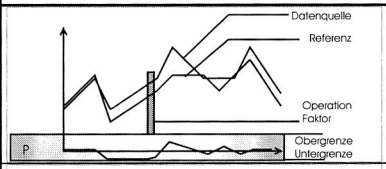
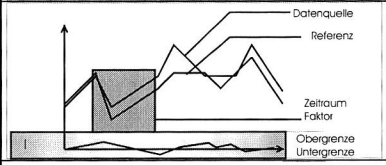
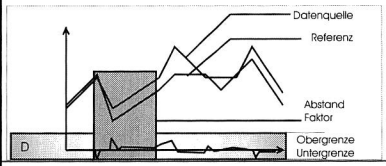
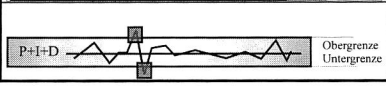
Anteil	Arbeitsweise	Parameter
P		Datenquelle (Ist) Referenz (Soll) Operation Faktor Obergrenze Untergrenze
I		Datenquelle (Ist) Referenz (Soll) Zeitraum Faktor Obergrenze Untergrenze
D		Datenquelle (Ist) Referenz (Soll) Zeitraum Faktor Obergrenze Untergrenze
Gesamt		Referenz (Soll) Obergrenze Untergrenze

Tabelle 5: Struktur, Arbeitsweise und notwendige Parameter der Störungserkennungskomponenten auf der Basis von PID-Komponenten

7.1.3 Formalisierung und Konzeption der Störungserkennung

Nachdem das grundlegende Konzept zur Störungsbeschreibung entwickelt ist, kann nun die Konzeption der Störungsregeln durchgeführt werden. Hierzu werden zunächst die erarbeiteten Störungsbeschreibungen in eine allgemeine Beschreibungsform (EBNF: erweiterte Backus-Naur-Form) übertragen:

```
Verlauf     ::= (Sollwert, Istwert, AnteilP,
                 AnteilI, AnteilD, Summe)

AnteilP     ::= Operation, Faktor, Grenzen

AnteilI     ::= Zeitraum, Faktor, Grenzen

AnteilD     ::= dt, Faktor, Grenzen

Summe       ::= Grenzen

Grenzen     ::= Obergrenze, Untergrenze
```

```
Zeitraum    ::= Offset_Anfang, Offset_Ende

Operation   ::= '-' | '/'
```

Damit diese Beschreibung zu einer Regel erweitert werden kann, muss die Verknüpfung des Verlaufs an Aktionen für den Abweichungs- bzw. Störungsfall hergestellt werden. Als Aktionen sind verschiedene Optionen denkbar:

- Zunächst kann eine Aktion einen textuellen Hinweis für den Benutzer erzeugen, der daraufhin individuell geeignete Maßnahmen ergreift. Dies kann immer dann zweckmäßig sein, wenn es sich um nicht näher definierbare Abweichungen oder um eine Situation handelt, in der sich eine Störung erst abzeichnet und durch prozessnahe Maßnahmen Abhilfe geschaffen werden kann.
- Weiterhin müssen u.U. bei einer Störung die betroffenen Organisationseinheiten benachrichtigt werden, damit weiterreichende Aktionen initiiert werden können.
- Zuletzt ist es auch sinnvoll, nach dem Erkennen einer Störung weitere Programme anzustoßen (z.B. eine Diagnose), damit hierdurch eine detailliertere Analyse der Situation durchgeführt werden kann (dies ist Gegenstand von Kap. 8, in dem auf dieser Grundlage ein simulationsbasiertes Monitoring entwickelt wird.) Es ist daher anzustreben, wiederum die Simulation als Analysewerkzeug heranzuziehen, um die dynamischen Effekte einer Störsituation effizient untersuchen zu können.

Die Störungsregeln werden wie folgt formalisiert und zur Implementierung in ein relationales Datenbankschema umgesetzt:

```
Regel       ::= [Element, Verlauf, Aktion, Information]+

Verlauf     ::= Referenz

Information ::= Referenz

Aktion      ::= "Textfeld" |
                [Organisationseinheit]* |
                [Programm ( [Element]+ ) ]*

Element     ::= Sollwert, Istwert

Sollwert    ::= Referenz

Istwert     ::= Referenz |
                Bezeichner im Simulationsmodell |
                Bezeichner im Prozess
```

```
Referenz    ::= IOR |
                Abfrage |
                Konstante
```

7.2 Realisierung der Überwachungsbausteine

Die Umsetzung der Störungsauswertung muss unter Berücksichtigung des modularen Gesamtaufbaus des Systems in mehreren Schritten erfolgen.

Vorbereitend sind zunächst einige theoretische Vorarbeiten zu leisten, die die Abtastung von Werten genauer untersuchen. Da die Abtastung in diesem Fall auch als eine Transformation von zeitkontinuierlichen in zeitdiskrete Prozessgrößen aufgefasst werden muss, sind Methoden der Wahrscheinlichkeitstheorie anzuwenden, mit denen eine möglichst rechenzeitökonomische Abtastrate Δt bestimmt werden soll (Kap. 7.2.1). Es soll erreicht werden, dass möglichst alle relevanten Ereignisse (signifikante Veränderung von Kenngrößen) erfasst werden, ohne jedoch gleichzeitig unnötig viele Abtastungen vornehmen zu müssen.

Anschließend ist festzulegen, wie die Verfügbarkeit der zu überwachenden Kennwerte sichergestellt werden kann. Dazu wird konzeptionell davon ausgegangen, dass alle Kennwerte im Simulationsmodell verfügbar sind. Diese Forderung kann entweder dadurch erfüllt werden, dass die Kenngrößen aus dem Simulationslauf heraus erfasst oder durch Ankopplung von Proxy-Objekten aus dem laufenden Betriebsgeschehen akquiriert werden. In beiden Fällen kann die Weiterverarbeitung der o.g. Überwachungsregeln dann im Simulationssystem vorgenommen werden.

Dieser Weg wird gewählt, weil es hierdurch vereinfachend möglich ist, die gesamte Auswertung des Simulationssystems zu realisieren. Konzeptionell wäre es dabei ebenso einfach, die Auswertung der Überwachungsregeln in einem eigenen Funktionsblock von geschachtelten Wrappern vorzunehmen und nur noch die daraus resultierenden Ereignisse an die Simulation zu melden.

Die Integration in den Simulator hat den Vorteil, dass die Auswertungsbausteine nicht nur für eine Überwachung von Daten aus DISDACOS genutzt werden können. Vielmehr können die zu erstellenden Bausteine universell zur Überwachung von Simulationsmodellen eingesetzt werden.

Im letzten Schritt können dann die diskretisierten Prozessgrößen anhand der in Kap. 7.1.3 beschriebenen Überwachungsregeln ausgewertet werden. Dazu werden im Simulationsmodell entsprechende Bausteine erstellt (Kap. 7.2.2 bis 7.2.5).

7.2.1 Konzeption und Realisierung des Abtastmoduls

Aufgabe des Abtastmoduls ist die Überwachung eines Parameters auf die Einhaltung vorgegebener Grenzen. Daraus ergibt sich für das Abtastmodul primär die Aufgabe, solche zeitraumbehafteten Werte periodisch zu erfassen. Alle zeitpunktbehafteten Werte (dies sind Ereignisse, z.B. durch Sensoren ausgelöst) werden zweckmäßigerweise direkt durch das Simulationssystem, durch Datenbanktrigger oder Wrapperfunktionen erzeugt und über die Eventchannel asynchron der Architektur gemeldet.

Prinzipiell wäre dazu bei einem zeitdiskreten Simulationssystem eine hohe Abtastrate möglich. Dies ist jedoch nicht zweckmäßig, da reale technische Prozesse meist um ein Vielfaches langsamer ablaufen. In der Simulation würden sich die Werte dabei während vieler Abtastperioden nicht ändern und somit auch keine neuen Informationen liefern (Überabtastung). Zudem würde bei größeren Modellen mit vielen Überwachungskomponenten die Simulationsgeschwindigkeit durch die unnötigen Rechenoperationen deutlich vermindert werden. Die zweckmäßige Wahl der Abtastperiode Δt und der Überwachungszeitspanne T, für deren Dauer das Abtastmodul aktiv ist, sind deshalb von Fall zu Fall unterschiedlich und müssen vom Anwender an die jeweiligen technischen Gegebenheiten der realen Fertigung angepasst werden. An dieser Stelle sollen zur Wahl der Abtastperiode noch zwei weiterführende Betrachtungen angestellt werden. Zum einen wird dabei von Ereignissen ausgegangen, die in konstanten zeitlichen Abständen auftreten: Hierbei ist die Abtastrate gleich dem Ereignistakt zu wählen. Zum anderen wird die Möglichkeit einer Wahrscheinlichkeitsbetrachtung angenommen, die für die Ereigniszeitpunkte eine statistische Verteilung zugrundelegt. Hier wird (vereinfachend) davon ausgegangen, dass viele Zufallsvariablen, die bei Experimenten und Beobachtungen in der Praxis auftreten, normalverteilt sind [144]. Ihre Dichtefunktion wird durch

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

mit μ Mittelwert und σ Standardabweichung beschrieben.

Ziel ist es nun, die Abtastperiodenlänge t so zu wählen, dass mit einer bestimmten Wahrscheinlichkeit γ in jedem Arbeitszyklus (Periode, in der ein Ereignis auftritt) mindestens eine Abtastung erfolgt. Damit soll eine Überabtastung verhindert werden.

Mit den Regeln der Wahrscheinlichkeitsrechnung, z.B. nach [145], lässt sich die gesuchte Abtastperiodenlänge t berechnen. Es wird gewährleistet, dass mit der vorgegebenen Wahrscheinlichkeit γ innerhalb jedes Bearbeitungszeitintervalls t_b eine Abtastung erfolgt. Es soll gelten:

$$P(\Delta t_b > \Delta t) \geq \gamma$$

und damit $P(\frac{\Delta t_b - \mu}{\sigma} > \frac{\Delta t - \mu}{\sigma}) \geq \gamma \Rightarrow 1 - \Phi(\frac{x - \mu}{\sigma}) \geq \gamma \Leftrightarrow \Phi(\frac{x - \mu}{\sigma}) \leq 1 - \gamma$

Hierbei ist $\Phi(\frac{x - \mu}{\sigma}) = F(x)$, was nach [146] gilt und zu $z = \frac{\Delta t - \mu}{\sigma} \Leftrightarrow \Delta t = z\sigma + \mu$ umgeformt werden kann. z wird dann numerisch gelöst, was durch ein einfaches DDE-gekoppeltes Basicscript in MS-Excel erfolgt.

Beispiel: Mittlere Bearbeitungszeit einer Drehmaschine: 60 Sekunden $\Rightarrow \mu$=60
Standardabweichung σ=10;
gewählte Wahrscheinlichkeit γ=90% $\Rightarrow$1-γ = 0,1
$\Rightarrow$ mit den tabellierten Werten, z.B. nach [147]: z = -1,282
$\Rightarrow$ mit o.g. Gleichung: $\Delta t \approx 47$ (Sekunden)

Das Abtastermodul ist so realisiert, dass die o.g. Gleichung bei Angabe der stochastischen Grunddaten automatisch ausgewertet und das Abtastmodul parametriert wird. Die in den folgenden Kapiteln beschriebenen Komponenten bauen darauf auf.

Eine weitere Aufgabe des Abtastmoduls ist die effiziente Speicherung der erfassten Werte über einen gewissen Zeitraum in einem Ringpuffer. Dies ist notwendig, um einerseits Auswertungen über mehrere Abtastperioden zu ermöglichen, z.B. für Mittelwertbildungen oder Summationen. Andererseits können die gespeicherten Werte als Historie zu Auswertungszwecken herangezogen werden.

7.2.2 Konzeption und Realisierung der P-Komponente

Die Aufgabe der P-Komponente (Proportional-Komponente) ist es, den Wert einer Modell- oder Datenquellengröße während der vom Anwender eingestellten Zeitspanne auf die Einhaltung vorgegebener Toleranzgrenzen zu überwachen.

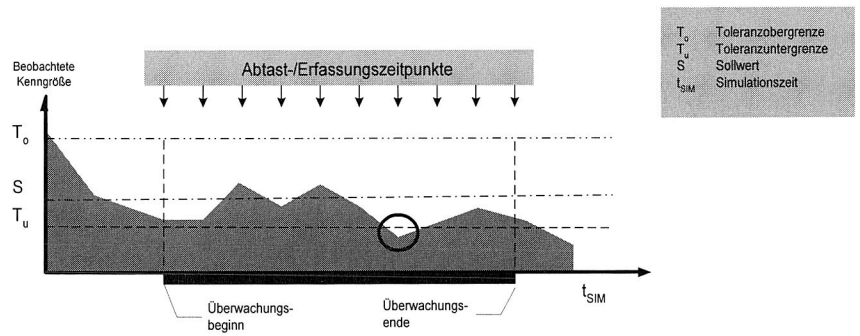


Abb. 84: Parametrierung der P-Komponente mit Toleranzober- und untergrenze, Überwachungsperiode mit äquidistanten Erfassungszeitpunkten

Hierbei können sowohl ein Sollwert, als auch die zulässige Ober- und Untergrenze unabhängig voneinander festgelegt werden. Während der Überwachungszeit wird jeweils im Abstand Δt der Abtastperiodenlänge ein Wert erfasst, gespeichert und gegen die Toleranzgrenzen getestet. Abb. 84 verdeutlicht dies.

Die Auswertung der Vorgaben erfolgt nach jedem Abtastvorgang. Die Überwachung der Über-/Unterschreitung soll jedoch nur dann eine Meldung auslösen, wenn sich unter Berücksichtigung der Parameter der Zustand der Datenquelle vom Zustand 'normal' in einen Zustand 'Abweichung' ändert, d.h. bei Überschreiten einer Grenze wird die zugehörige Regel ein einziges Mal ausgeführt. Die nächste Meldung wird erst dann erzeugt werden, wenn entweder eine externe Störung eintritt oder ein Zustandsübergang überschritten→normal erfolgt ist. Diese Zustandsübergänge von 'innerhalb' nach 'außerhalb' des zulässigen Toleranzbereichs sollen als Trigger bezeichnet werden. Abb. 85 zeigt dies als Zustandsautomat.

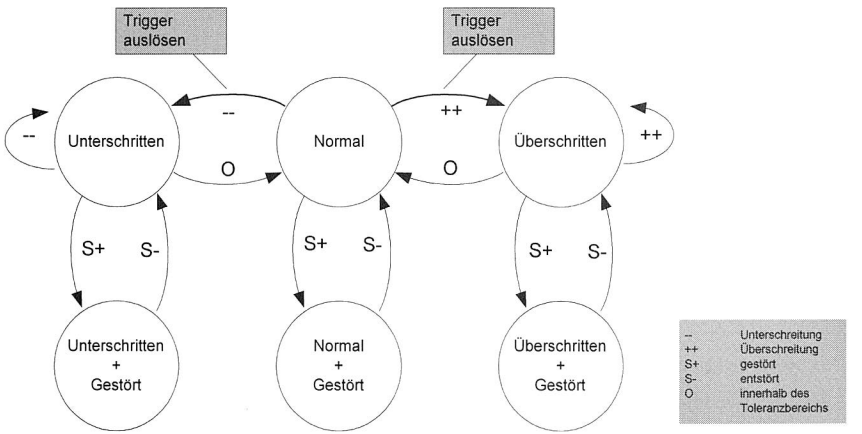


Abb. 85: Mögliche Zustände einer Überwachungskomponente einschließlich der Übergänge, die als Trigger für eine Meldung verwendet werden

Wird ein Trigger ausgelöst, können gemäß der Spezifikationen aus Kapitel 7.1.3 verschiedene Aktionen angestoßen werden.

- Der beobachtete Baustein wird farblich gekennzeichnet.
- Das Ereignis wird in einer Tabelle protokolliert.
- Das Störereignis wird mit Ort, Zeit und vollständigem Historiendatensatz über den DISDACOS-Event-Channel an eine andere Anwendung geschickt.
- Es wird ein Nachrichtenfester eingeblendet.
- Die Simulation wird angehalten.

- Es wird ein Methodenfenster geöffnet, in dem der Benutzer direkte Eingaben machen kann (beispielsweise eine Entstörmaßnahme aktivieren).
- Eine Störung aufgrund einer Parameterabweichung wird in eine Bausteinstörung umgesetzt. Dadurch kann beispielsweise der Materialfluss im Simulationsmodell unterbrochen werden.

Alle genannten Aktivitäten lassen sich beliebig kombinieren und in einer Eingabemaske zusammen mit allen anderen Parametern einstellen. Abb. 87 zeigt dies.

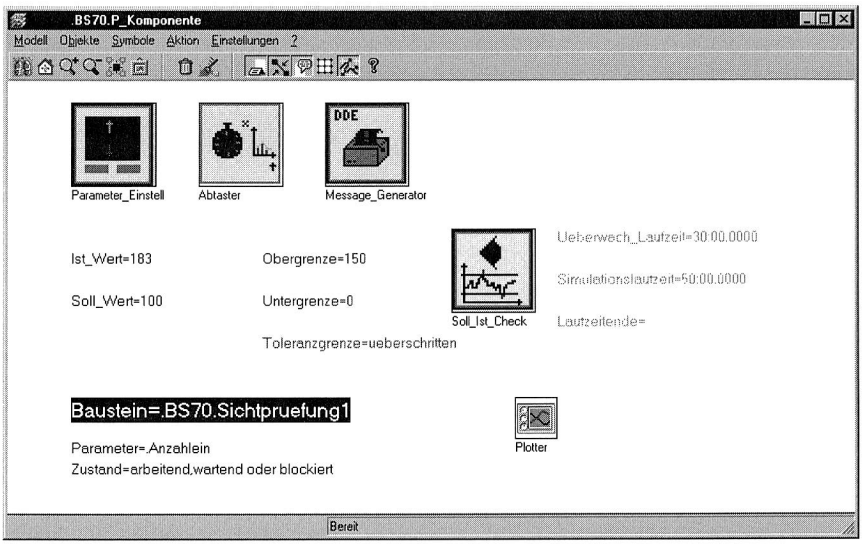


Abb. 86: Aufbau der P-Komponente mit den Subkomponenten zur Datenerfassung, -auswertung, Meldungserzeugung und Anzeige im Simulator eM-Plant

Die P-Komponente ist in Ergänzung der Datenkopplungsbausteine ebenfalls konzipiert und im Simulationssystem eM-Plant (vormals Simple++) nach den Maßgaben aus [82] als Referenzbaustein realisiert worden. Abb. 86 und Abb. 87 zeigen den vollständigen Aufbau der P-Monitoringkomponente. Der Einsatz wird in Kap. 7.3 vorgestellt.

Dieser Baustein kann in beliebige Simulationsmodelle eingesetzt werden, ohne dass neben den Parametrierungen weitere Anpassungen vorgenommen werden müssen.

7.2.3 Konzeption und Realisierung der I-Komponente

Die Funktionsweise der Integrationskomponente (I-Komponente) ähnelt konzeptionell zunächst der P-Komponente. Alle Eigenschaften des Abtasters und weitgehend alle Teile der Parametrierung bleiben unverändert.

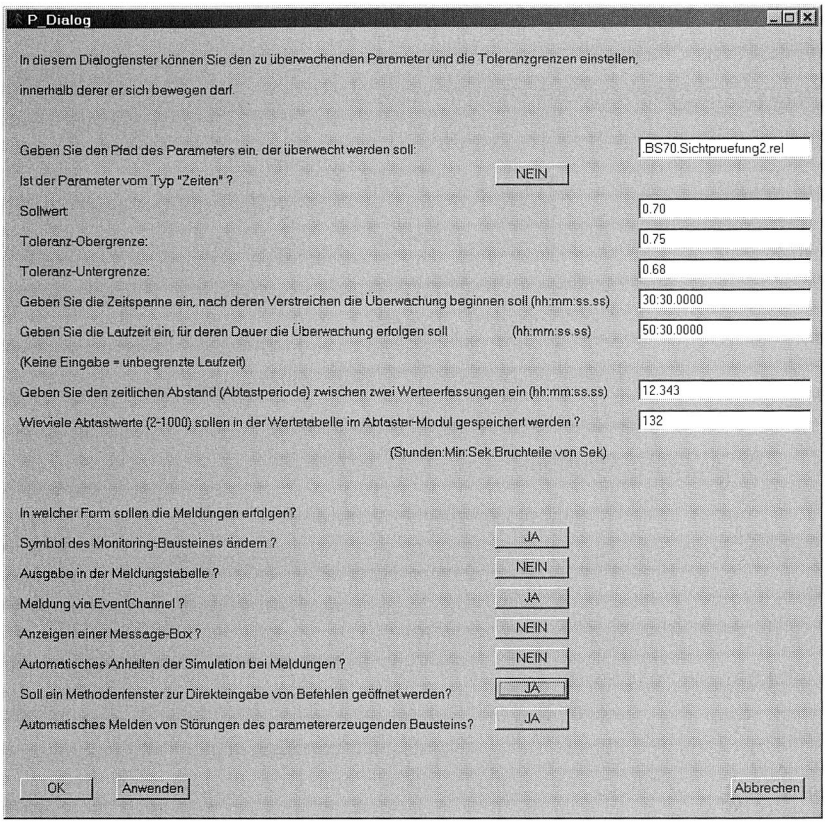


Abb. 87: Benutzerdialog der P-Komponente zur Parametrierung der Abtast-, Überwachungs- und Meldungsfunktionen dieses Bausteins im Simulationssystem

Ergänzt wird die realisierte Komponente durch eine umfangreiche Subkomponente zur Realisierung der 'Integration'. Auch hier ist zunächst die Diskretisierung, d.h. die Transformation der Integration in eine Summation, durchzuführen. Dabei sind jedoch einige Punkte zu beachten, die sich aus den praktischen Anforderungen an eine solche Komponente ergeben:

Die Summation der einzelnen Elemente der Messwertfolge ist beschränkt auf eine vom Benutzer definierbare Periodenanzahl innerhalb eines Bewertungsfensters. Diese Beschränkung wird genutzt, um das 'Gedächtnis' des Auswerters mit einem Zeithorizont zu versehen. Dies ist wichtig, um eine Abweichung charakterisieren zu können, beispielsweise "Wenn der Pufferbestand während der letzten zwei Stunden häufig über seinem Sollbestand lag, kann dies als ein Hinweis auf einen sich ab-

zeichnen Wartungsbedarf an der folgenden Drehzelle gewertet werden. Deshalb ist ein Hinweis zu generieren."

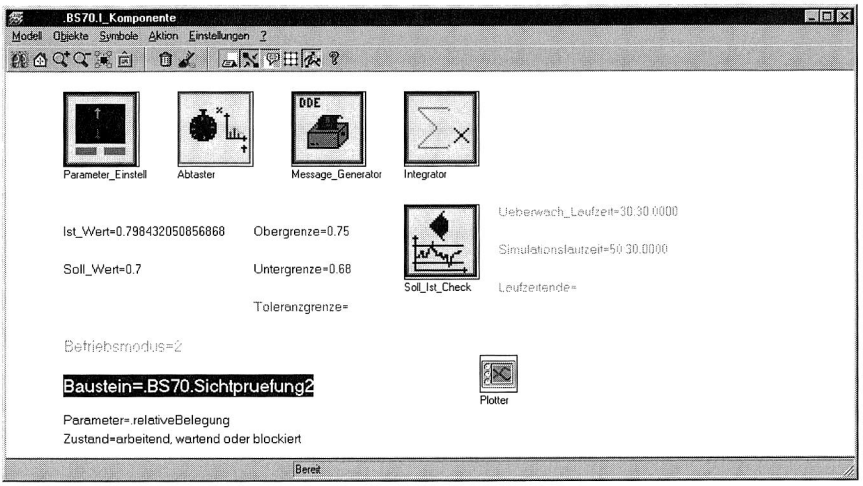


Abb. 88: Aufbau der I-Komponente mit den Subkomponenten zur Datenerfassung, -auswertung, Meldungserzeugung und Anzeige im Simulator eM-Plant

Die Flächen Δt * *(Ist_Wert - Soll_Wert)* gehen bei der Summation klassischerweise betragsrichtig in die Berechnung ein. Da sowohl positive wie negative Flächeninhalte als Rechengrößen auftreten, kann sich bei betragsmäßig gleichen Flächen die Gesamtfläche 0 ergeben. Dies soll als Betriebsmodus 1 der I-Komponente bezeichnet werden. Als Modus 2 wird dementsprechend eine Betriebsart bezeichnet, bei der die Über-/Unterschreitung des Sollwerts getrennt aufsummiert wird und unabhängig voneinander bewertet wird. Dies ist beispielsweise hilfreich, um einen länger anhaltenden Pufferbestand in der Nähe des Auslösebestandes (Untergrenze) zu erkennen.

7.2.4 Konzeption und Realisierung der D-Komponente

Eine Differential-Komponente (D-Komponente) wird dazu verwendet, die zum aktuellen Abtastzeitpunkt vorliegende Soll-Ist-Wert-Differenz $x_D(t)$ in Relation zu einem (nicht notwendigerweise unmittelbaren) Vorgängerwert $x_D(t-i\Delta t)$ zu setzen. Mathematisch kann diese Veränderung durch die Bildung der ersten zeitlichen Ableitung beschrieben werden: Δx_D ist ein Maß für die Steigung bzw. für das Gefälle zwischen zwei x_D-Werten.

Der Einsatzzweck liegt in der Erkennung von sprunghaften oder schnellen Veränderungen des Ist-Werts innerhalb einer definierbaren Zeitspanne. Dies kann bei der Er-

kennung von Störungen hilfreich sein, bei denen beispielsweise der Bestand eines Puffers schneller als gewünscht steigt, weil eine nachfolgende Station gestört ist.

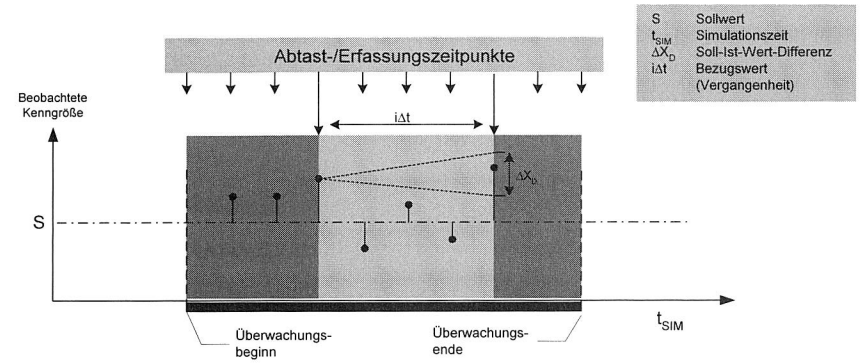


Abb. 89: Funktionsweise und wichtige Parameter der D-Komponente mit maximal zulässiger Differenz und Bezugswert

Im Unterschied zur P-Komponente werden keine Absolutwerte als Toleranzgrenzen vorgegeben, sondern Werte für die maximal zulässige positive bzw. negative Veränderung innerhalb einer festen Simulationszeitspanne. Die grundsätzliche Funktionsweise fasst Abb. 89 nochmals zusammen.

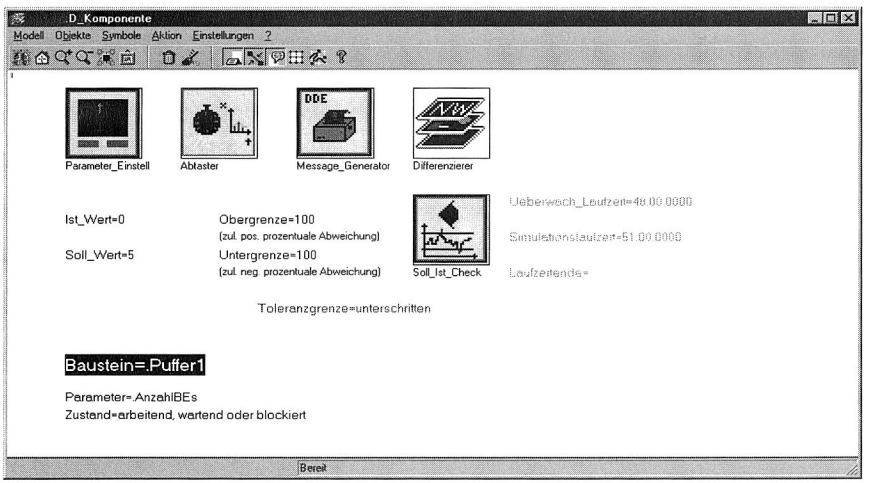


Abb. 90: Implementation der D-Komponente als Referenzbaustein im Simulationssystem eMPlant.

Die Abgrenzung zur I-Komponente besteht darin, dass diese die Soll-Ist-Wert-Abweichungen kumuliert über einen Bewertungszeitraum betrachtet, wohingegen bei

der D-Komponente immer die Parameterwerte zu zwei exakt definierten Zeitpunkten verglichen werden. Die realisierte D-Komponente ist ebenfalls ein Referenzbaustein, der mit dem Simulationssystem eM-Plant realisiert wurde. Abb. 90 zeigt die Implementation des Bausteins.

7.2.5 Konzeption und Realisierung der Linear-Komponente

Die Linearkomponente ergibt sich aus der Addition der Ergebnisse der P-, I- und D-Komponenten. Deshalb ist die Linearkomponente als P-Komponente mit Additionsfunktion aufzufassen. Die Beschreibung ist analog zu Kap. 7.2.2.

7.3 Einsatzbeispiel

Gegenstand des folgenden Beispiels ist die Abbildung eines bestehenden Montagesystems für Anti-Blockier-Systeme (ABS) eines Automobilzulieferers. Mittels dieses Modells sollte eine Schwachpunktidentifikation der bestehenden Situation durchgeführt werden. Darauf aufbauend waren Verbesserungsmaßnahmen abzuleiten und zu bewerten.

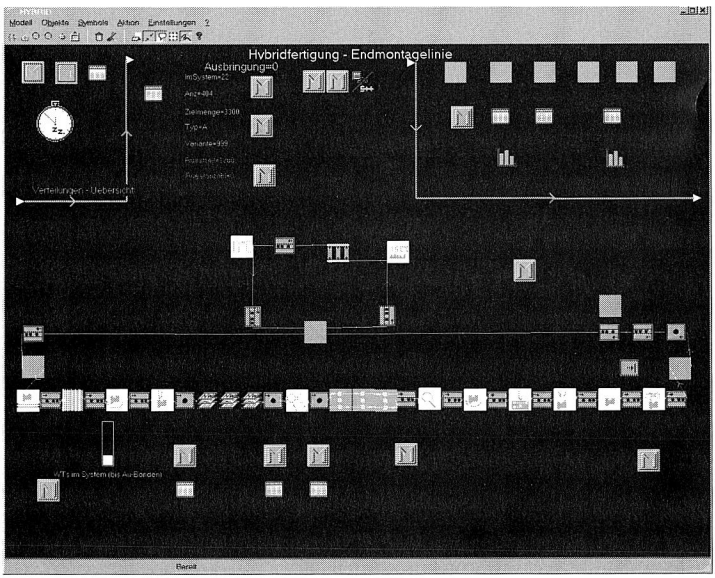


Abb. 91: Simulationsmodell einer ABS-Hybridmontage im Linienaufbau im Simulationssystem eM-Plant

Für die Modellierung dieses Montagesystems waren zunächst die erforderlichen und verfügbaren Daten wie zum Beispiel Taktzeiten, Verfügbarkeiten, Auftragsdurchläufe,

Arbeitszeitmodelle, Strukturinformationen und Strategien zu erheben und in ein Simulationsmodell umzusetzen. Abb. 91 zeigt das erstellte Simulationsmodell. Die im vorliegenden Simulationsmodell wie auch in der Realität zu beobachtende Situation war geprägt von einem inhomogenen Materialfluss, der sich im wesentlichen durch stark streuende Bearbeitungs- und Störzeiten manifestierte und auf punktuelle Schwachpunkte im Materialfluss zurückgeführt werden konnte.

Bei der simulationsgestützten Ursachenidentifikation wurden zunächst in der üblichen Weise Simulationsexperimente durchgeführt. Dabei wurde ausgehend von einer durch den Experimentplan vorgegebenen Initialisierung des Modells (Zeiten, Systemlast, Layout) eine Woche realer Produktionszeit vollständig in der Simulation durchlaufen. Anschließend waren die Ergebnisse auszuwerten. Da es sich bei dem System um eine teilelastisch gekoppelte Linie handelte (Linie mit sehr geringen Pufferkapazitäten zwischen einzelnen Stationen), waren zur Identifikation geeigneter Verbesserungen meist mehrere Experimente notwendig (Problem des wandernden Flaschenhalses). Durch die Vielzahl der gleichzeitig im Modell befindlichen passiven, beweglichen Objekte (zwischen 850 und 1000 Objekte) belief sich die Rechenzeit eines Simulationsexperiments auf ca. 50 bis 70 Minuten.

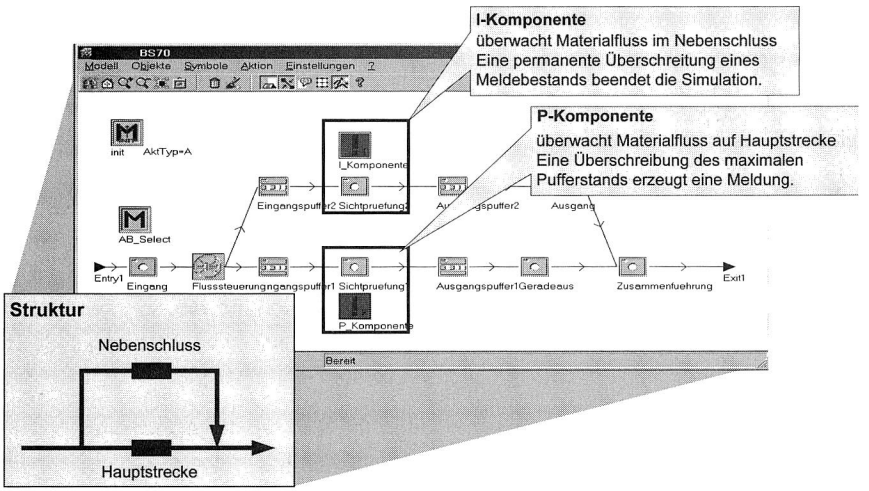


Abb. 92: Nutzung eines P- und I-Bausteins zur Überwachung des Materialflusses

In diesem Modell konnte die Funktion und Leistung der Monitorkomponenten innerhalb eines Simulationsmodells 'stand alone' bewertet werden. Hierzu wurden geeignet parametrierte P-, I- und/oder D-Komponenten an Stellen im Modell eingesetzt, die als kritisch angesehen wurden. Dies waren Punkte im System, bei denen nach den Betriebserfahrungen des Betreibers offensichtlich Probleme im Materialfluss o-

der der Materialbereitstellung vorherrschten. Ziel war es, bereits während der Simulationslaufzeit auf inakzeptable Abweichungen (Störungen nach Definition aus Kap. 7.1.1) hinzuweisen und ggf. das Simulationsexperiment frühzeitig abzubrechen. Dadurch wurde eine erhebliche Zeitersparnis erreicht, weil nach Eintritt eines kritischen Systemzustandes der weitere Simulationslauf meist uninteressant war. Im Beispiel waren dies etwa permanent volle/leere Puffer, stark streuende Durchlaufzeiten oder schnell steigende Nebenzeitanteile.

Die Monitoringkomponenten wurden als Referenzbausteine in das Simulationsmodell eingesetzt. Abb. 92 verdeutlicht dies. Die entsprechenden Parameter für Sollwerte und Toleranzgrenzen konnten durch die Betriebserfahrung des Linienpersonals und der Produktionsplaner problemlos ermittelt werden. Im Beispiel wurden Monitorkomponenten verwendet (P und I), mit denen zwei materialflussentkoppelnde Puffer des Produktionssystems überwacht wurden.

Der P-Monitor (unten im Bild) wird zur Überwachung der Hauptstrecke im Materialfluss verwendet. Die Parametrierung ist so gewählt, dass der absolute Pufferbestand einen Wert von 20 Werkstückträgern nicht überschreiten soll. Damit wird auf einfache Weise geprüft, ob durch Störungen im folgenden Materialflusssystem ein Rückstau entsteht. Eine Übersicht der eingestellten Parameter gibt Tabelle 6.

Im Nebenschluss wird ein I-Monitor genutzt, um die Pufferfüllung vor der eigentlichen Bearbeitungsstation zu überwachen. Der Integrator wird in diesem Fall verwendet, um eine länger dauernde Über- oder Unterschreitung eines sinnvollen Pufferbestandes zu überwachen. Eine längere Unterschreitung des Zielbestandes würde darauf hinweisen, dass im vorgelagerten Teil des Systems ein Engpass zu vermuten ist, der die Homogenität des Materialflusses stört und demzufolge die Auslastung der Bearbeitungsstation aufgrund von Wartezeiten zu gering ist. Demgegenüber würde eine längere Überschreitung des Zielbestandes an dieser Stelle auf eine falsche Abstimmung von Materialfluss und Bearbeitungsstation hinweisen, was - bedingt durch verlängerte Liegezeiten - auch zu Qualitätsproblemen führen könnte.

Durch den Einsatz der Monitoringkomponenten in diesem Simulationsmodell konnte bei etwa 50% der Experimente der Simulationslauf bereits nach etwa 15-20% der Gesamtsimulationszeit abgebrochen werden, da sich 'signifikante' Abbruchkriterien meist kurz nach der Einschwingphase abzeichneten.

Die Zeitersparnis durch die verringerte Laufzeit konnte anschließend für weitere Experimente und eine intensivere Ergebnisauswertung genutzt werden. Speziell der zweite Punkt wurde durch die Monitoringkomponenten unterstützt, da nicht nur die Tatsache des frühzeitigen Abbruchs, sondern auch die bereitgestellten Verlaufsdaten

weitere Erkenntnisse lieferten. Dadurch konnten Verbesserungsmaßnahmen schneller gefunden werden.

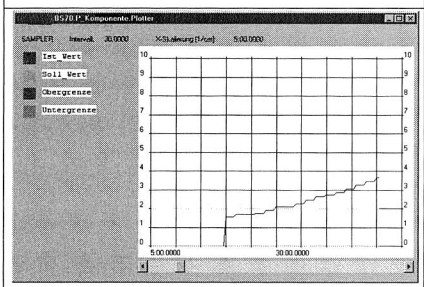
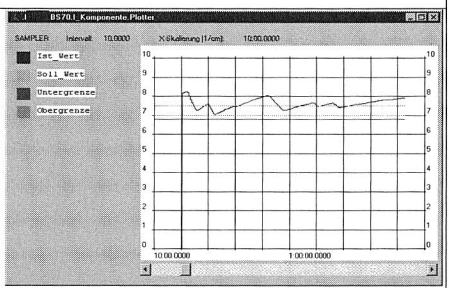
Parameter des P-Monitors		**Parameter des I-Monitors**	
Überwachter Parameter:	**.Montage.BS70 .Sichtpruefung1 .anzahlBEs**	Überwachter Parameter:	**.Montage.BS70. Sichtpruefung2. relativeBelegung**
Parameter vom Typ Zeiten?	**[NEIN]**	Parameter vom Typ Zeiten?	**[NEIN]**
Anhalten bei Überschreitung?	**[JA]**	Anhalten bei Überschreitung?	**[JA]**
Sollwert:	**10**	Sollwert:	**0,70**
Obergrenze:	**20**	Obergrenze:	**0,75**
Untergrenze:	**0**	Untergrenze:	**0,68**
Startzeit:	**20:00.00**	Startzeit:	**20:00.00**
Laufzeit:	**8:00:00.00**	Laufzeit:	**8:00:00.00**
Abtastperiode:	**1:43.23**	Bewertungsintervall-Länge:	**4:17.87**
Anzahl der gespeicherten Werte:	**150**	Abtastungen pro Intervall:	**10**
		Anzahl der gespeicherten Werte :	**10**
Kennwertverlauf während der Simulation		**Kennwertverlauf während der Simulation**	
			

Tabelle 6: Parametrierung des P- und I- Monitors am Beispiel einschließlich des Kennwertverlaufs während eines Simulationsexperiments

7.4 Zusammenfassung

Zur Überwachung von online-Datenquellen wurden in Anlehnung an die aus der Regelungstechnik bekannten PID-Regler Monitorkomponenten entwickelt, die auf der Basis von parametrierbaren Bausteinen die Sollwertabweichungen von Kenngrößen in zeitdiskreten Simulationsmodellen erfassen. Über benutzerdefinierbare Auslöseschwellen können signifikante Abweichungen (Störungen) erkannt werden, die an-

schließend weitere Aktionen innerhalb des Simulationssystems veranlassen können. Dieses Konzept wurde anschließend exemplarisch mit dem Simulationswerkzeug Simple++ (eM-Plant) realisiert.

Die Bewertung von Funktion und Leistungsfähigkeit der Monitorkomponenten innerhalb eines Simulationsmodells konnte anhand eines Simulationsprojekts im industriellen Umfeld erfolgreich nachgewiesen werden. Zudem ergab sich bei reduzierten Simulationslaufzeiten ein verbessertes System- und Parameterverständnis. Es konnte gezeigt werden, dass eine umfassendere Experimentauswertung auf der Basis der einfach zu parametrierenden Bausteine erreicht werden konnte.

Als erweiterter Nutzen zeigte sich in Analogie zur üblichen Datenbeschaffung durch den Einsatz der P-,I- und D-Komponenten eine schnelle, strukturierte und sichere Abfrage wichtiger Systemcharakteristika, die Modell- und Ergebnisqualität erhöhen.

Im folgenden Kapitel werden diese Ergebnisse dazu verwendet, eine Laufzeitüberwachung von online-Daten aufzubauen, die Prognosen über die Auswirkung einer Störung auf das laufende Produktionssystem abgibt. Hierzu werden alle Systemteile dieser Arbeit - insbesondere die Funktionen von DISDACOS - in einem umfassenden Szenario vorgestellt.

8 Simulationsbasiertes online-Monitoring

Unter einem Monitoring wird im vorliegenden Zusammenhang die Leistungsbewertung und -überwachung sowie die Ablaufbeobachtung von Produktionssystemen verstanden. Dies ist bei Produktionssystemen meist eine Leitstandsfunktion.

Ausgangspunkt eines simulationsbasierten online-Monitorings ist der permanente Datenabgleich zwischen Anlage und Simulation (Abb. 93). Hierbei spielt die Simulation zunächst die Rolle eines Leistungs- und Ablaufbeobachtungssystems (Monitors). Die Stärken der Simulation sind dabei aus verschiedenen Gründen nützlich:

- Es stehen seit der Planung des betrachteten Systems häufig Simulationsmodelle zur Verfügung. Das Simulationsmodell als Abbild des realisierten Planungsstands stellt bereits die betrieblichen Abläufe dar. Es ergibt sich damit also eine wirtschaftlich sinnvolle Weiterverwendung des Modells.
- Die Simulation kann parallel zur "Realität" auch in "Was-Wäre-Wenn"-Szenarien eingesetzt werden, beispielsweise bei der Optimierung der Auftragseinlastung [148].
- Die Simulation kann auf der Basis des aktuellen Datenmaterials Abweichungen und Störungen bezüglich definierter Soll-Abläufe visualisieren und bei Bedarf Alarm auslösen. Gleichzeitig kann über die Auswirkungen informiert und es können Reaktionsmöglichkeiten vorgeschlagen werden.

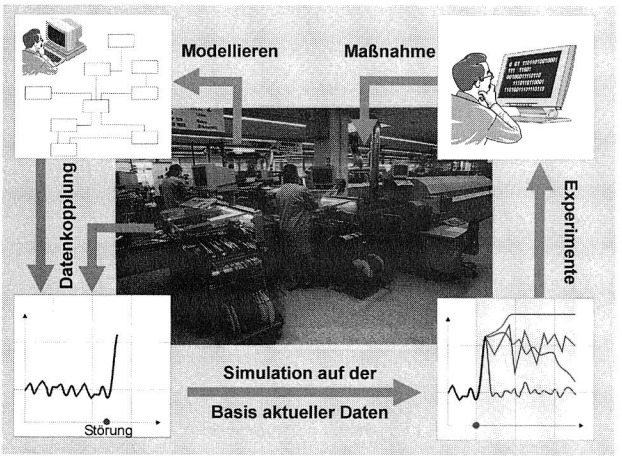


Abb. 93: Einsatz eines simulationsbasierten online-Monitorings mit integrierter Überwachungs- und Bewertungsfunktion

Die ersten beiden Punkte stellen -abgesehen von der Verwendung aktueller betriebsnaher Daten- bereits heute den Stand der Technik dar. Sie werden häufig durch Integration der Simulation in PPS-Systeme oder durch Zugriffe auf Datenbanken realisiert [149]. Die integrierte Monitoringfunktion mit Alarmmodulen wurde bisher jedoch in der Praxis kaum eingesetzt, da für deren umfassenden Einsatz entweder die Datenqualität nicht ausreichte oder die Latenzzeit bei der Datenakquisition zu lang war [62]. Für das Einsatzgebiet der Simulation als erweiterte Leitstandsfunktion sind deshalb noch wenig Vorgehensweisen und Werkzeuge bekannt, die tatsächlich ein 'dynamisches' Simulationsmodell, anstatt einer fortgesetzten Proberechnung bieten. Daher sind neben der eigentlichen Kopplung an die Architektur zusätzliche Methoden zu schaffen, die innerhalb eines Simulationsmodells eine Verarbeitung der Daten nach den o.g. Maßgaben erlauben. Dies betrifft hauptsächlich die Einbettung der neue Funktionalität zur Erkennung signifikanter Abweichungen (Kap. 7) in ein umfassendes Assistenzsystem.

8.1 Struktur des online-Monitoringsystems

Die Struktur des online-Monitoringsystems basiert auf den PID-Überwachungsbausteinen aus Kap. 7, eM-Plant zur Simulation, Steuerung und Visualisierung sowie MS-ACCESS als Datenbanksystem. Hieraus werden die Einzelmodule Kopplung/Verdichtung, Laufzeitüberwachung, Datenbank mit Fehlerbeschreibungen und Maßnahmen sowie Alarmmodul und Prognosemodul realisiert.

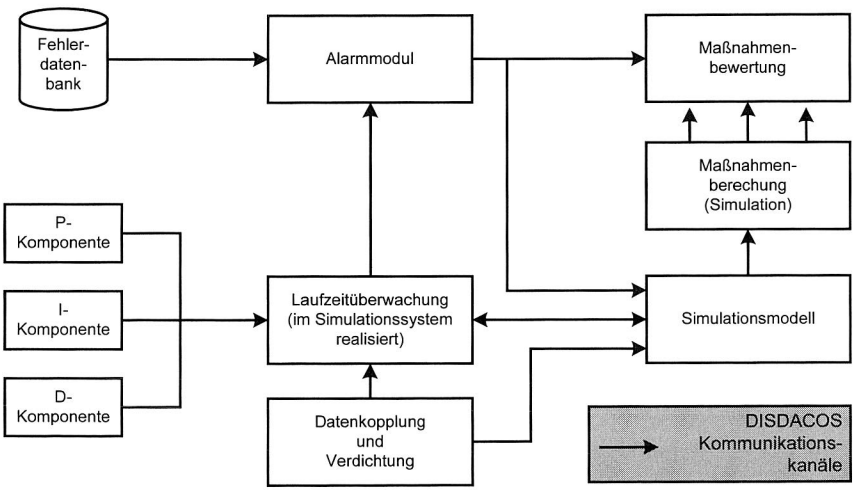


Abb. 94: Übersicht aller zum simulationsbasierten online-Monitoring notwendigen Module

Diese werden mit DISDACOS im Sinne eines Frameworks kommunikationstechnisch verbunden. Abb. 94 gibt einen Überblick über alle im folgenden beschriebenen Module.

8.1.1 Modul: Datenkopplung und Verdichtung

Ausgangspunkt der Laufzeitüberwachung ist ein 'klassisches' Simulationsmodell. Dies steht entweder aus der Planungsphase bereit und wird geeignet angepasst, oder es wird speziell für den Monitoringzweck erstellt.

Die Datenkopplung erfolgt hierbei, indem bei der Erstellung oder Erweiterung des Simulationsmodells dieses an DISDACOS angebunden wird. Hierzu wurden bereits in Kap. 5 die technischen Grundlagen vorgestellt. Somit werden relevante Datenquellen im Simulationsmodell als Bausteine dargestellt. Wie auf andere Variablen oder Attribute kann auf diese zur Simulationslaufzeit zugegriffen werden, d.h. durch Wrapper innerhalb der Architektur bereitgestellte und aktuelle Informationen sind innerhalb des Simulationsmodells verfügbar.

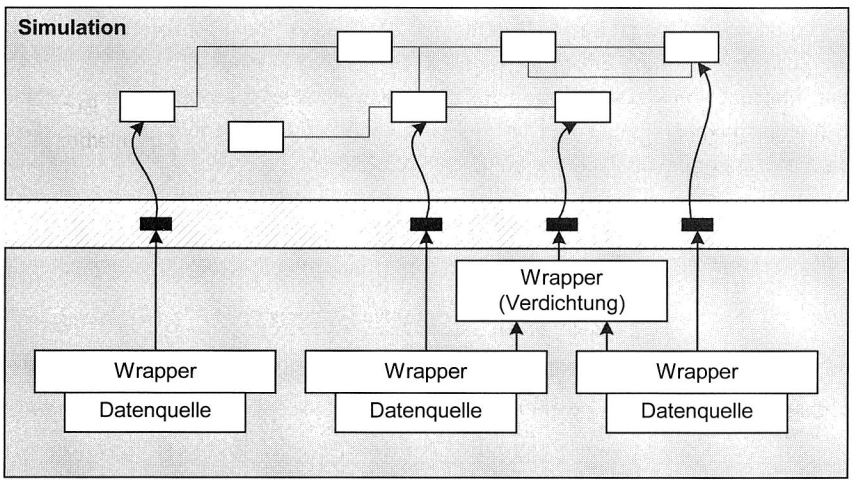


Abb. 95: Datenverdichtung durch Nutzung hierarchisch aufgebauter Wrapper

Innerhalb der Datenbereitstellung durch die Wrapper kann zusätzlich noch eine Datenverdichtung stattfinden. Hierbei werden Betriebsdaten zunächst durch einen oder mehrere Wrapper erfasst und anschließend von einem weiteren Wrapper verdichtet. Solche Kennwerte, beispielsweise Mittelwerte oder Durchlaufzeiten, stehen dann wiederum der Simulation zur Verfügung. Von dieser Möglichkeit der Architektur wird im Modul 'Datenkopplung und Verdichtung' intensiv Gebrauch gemacht (Abb. 95).

8.1.2 Funktion: Laufzeitüberwachung

Die Laufzeitüberwachung wird durch die Überwachungskomponenten aus Kap. 7 realisiert. Diese können synchron zur Simulationslaufzeit Fehler- und Störungsmeldungen generieren. Dazu werden die PID-Komponenten in das Simulationsmodell eingesetzt und parametriert. Ergänzend zum Einsatz in Kap. 7 muss zusätzlich der Baustein durch eine Methode in die Meldungskette eines dafür jeweils einzurichtenden Event-Channels integriert werden - dies ist die Schnittstelle zum Alarmmodul.

Die Synchronisation erfolgt durch die Möglichkeit des Simulationssystems, in "Echtzeitgeschwindigkeit" betrieben zu werden. Dabei ist die Simulationsgeschwindigkeit direkt an die reale Zeit gekoppelt, d.h. das Simulationsmodell läuft parallel zum Realsystem. Die hierdurch freiwerdende Rechenkapazität wird für die Auswertung der Überwachungsbausteine genutzt. Sollte diese Kapazität auf einem einzelnen Rechner nicht ausreichen, kann die Laufzeitüberwachung auch verteilt durch mehrere Rechner erfolgen. Dazu wird auf jedem Rechner das Monitoringmodell geladen und die Fähigkeit von DISDACOS ausgenutzt, alle Parameter in den Kommunikationsobjekten der Anwendungsschicht zwischenzuspeichern. Hierauf können dann alle Monitore zugreifen, was durch den Echtzeitbetrieb auch keine Konsistenzprobleme macht. Die Zusammenführung erfolgt im verteilten Betriebsfall in einer einzigen, gemeinsamen Meldungskette zum Alarmmodul. Im Meldungsfall wird schließlich eine Fehler-/Störungsnummer erzeugt und im Meldungskanal an das Alarmmodul übertragen. Die Struktur der Laufzeitüberwachung stellt Abb. 96 dar.

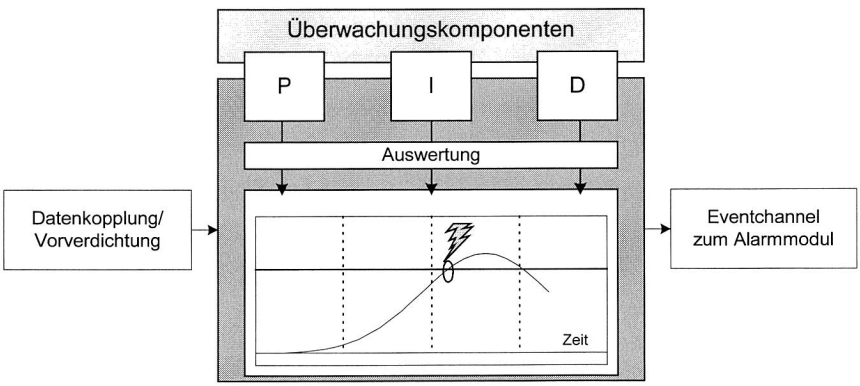


Abb. 96: Integration des Moduls "Laufzeitüberwachung" in die Monitoringkette

8.1.3 Modul: Datenbank mit Fehlerbeschreibungen und Maßnahmen

Eine Störungsmeldung kann entweder als asynchrone Meldung über den Eventchannel oder als synchroner Methodenaufruf über die Laufzeitüberwachung in das System gelangen. Gemeldet wird hierbei zunächst nur eine Fehlernummer. Um die-

ser nun einen Fehlerort sowie Sofort- und potentielle Gegenmaßnahmen zuordnen zu können, wird eine Datenbank verwendet, in der die hierzu notwendigen Informationen hinterlegt sind.

Folgende Informationen werden der Fehlernummer zugeordnet:

Ort: Textuelle Beschreibung des Ortes, an dem der Fehler / die Störung aufgetreten ist. Diese Zuordnung ist notwendig, da im Falle einer synchronen Meldung die auslösende Instanz nicht über das Navigationssubsystem in den Monitor gelangt und demzufolge auch nicht zwingend ein Navigationspfad (Ort) bereitgestellt werden kann. Um dennoch über eine eindeutige Ortsinformation zu verfügen, muss deshalb diese Information zusätzlich bereitgestellt werden. Darüber hinaus ist es denkbar, dass weitere Informationen bereitgestellt werden, die ausschließlich für die Fehler/ Störungsbehandlung notwendig sind, beispielsweise Multimediadaten wie Lagepläne, Zeichnungen usw.

Beschreibung: Da der Monitor als Assistenzsystem ausgelegt werden soll, muss für den Anwender eine detaillierte Beschreibung des Fehlers / der Störung bereitgestellt werden.

Priorität: Die Priorität einer Meldung legt deren Wichtigkeit fest, was im Falle einer Meldungslawine die Reihenfolge der Abarbeitung regelt.

Maßnahmen: Zur Störungsbehebung können Gegenmaßnahmen hinterlegt werden, die simulativ bewertet werden sollen. Eine 'Maßnahme' muss hierbei nicht zwangsläufig einer Einzelmaßnahme entsprechen. Vielmehr kann unter einer 'Maßnahme' auch ein 'Maßnahmenbündel' verstanden werden, welches mehrere Einzelmaßnahmen enthält. In der Datenbank wird die resultierende Maßnahmenliste verwaltet, wobei auch hier wieder eine Beschreibung in Textform, sowie bedarfsweise auch Programmmodule bereitgestellt werden.

Sofortmaßnahme: Da die Bewertung der Gegenmaßnahmen durch die Simulation zeitaufwendig ist (im später dargestellten Beispiel sind etwa 10 Sekunden Simulationszeit je Maßnahme zu erwarten), kann zunächst eine Sofortmaßnahme ergriffen werden. Da diese einerseits für den Anwender hilfreich sein kann, jedoch den Zustand des realen Systems zusätzlich verändern kann, muss diese separat aufgeführt und berücksichtigt werden.

Filter: Sie sind notwenig, da Fehler/Störungen häufig Folgefehler/-meldungen nach sich ziehen. Um solche Meldungslawinen hinsichtlich der Systemleistung unterbrechen zu können, werden Meldungsfilter vorgesehen, die die Weiterleitung spezifischer Meldungen blockieren, die nach einem Fehler/einer Störung auftreten können.

Weitere Informationen: Hier ist denkbar, beispielsweise Diagnoseinformationen verfügbar zu machen, wie etwa [150] in der Realisierung eines Expertensystems zur Telediagnose vorschlägt.

Der Zugriff auf die Datenbank wird über einen Wrapper abgewickelt, der die notwendigen Abfragen erzeugt, einen fehlerrelevanten Satz an Informationen zur weiteren Fehlerbehandlung erstellt und diesen im System bereitstellt.

8.1.4 Modul: Alarm

Das Alarmmodul koordiniert im Meldungsfall die weiteren Aktivitäten des Monitoringsystems. Hier werden die Meldungen ausgewertet, für den Benutzer in Klartext umgesetzt, ggf. eine Sofortmaßnahme eingeleitet und die Bewertung der Gegenmaßnahmen veranlasst.

Das Alarmmodul stellt einen Endpunkt in der Meldungskette, einen Consumer, bereit, an dem alle Meldungen auflaufen. Diese werden zunächst in einer Warteschlage zwischengespeichert, damit der Endpunkt für weitere Meldungen frei bleibt. Die Warteschlangeneinträge werden dann durch Anfragen an die Datenbank (Kap. 8.1.3) um die Informationen erweitert und gemäß der Prioritäten in eine weitere Warteschlange übernommen. Hierbei werden ggf. auch Meldungsfilter aktiviert, die die Meldung von Folgefehlern während der Alarmbehandlung unterbinden. Dies hat den positiven Effekt, dass die eintreffenden Informationen ausgedünnt werden, wodurch die Reaktionszeit des Gesamtsystems erhöht wird. Ebenso ist es möglich, für Diagnosezwecke eine gesamte Meldungslawine heranzuziehen, wodurch u.U. eine umfassende Fehlerklassifikation unterstützt wird (Abb. 97).

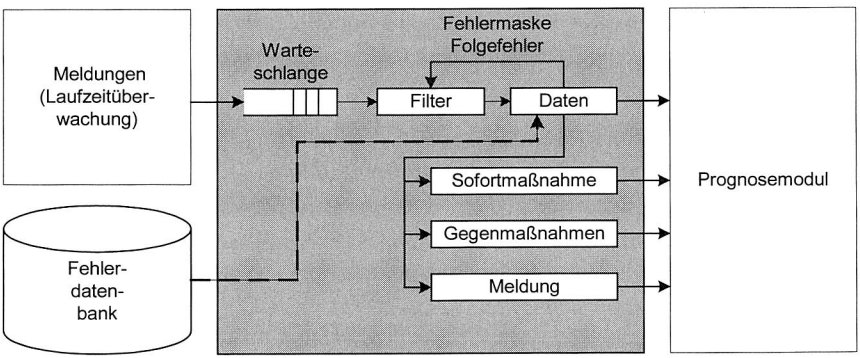


Abb. 97: Einbindung des Alarmmoduls in die Monitoringkette

Im weiteren werden die aus der Datenbank erhaltenen Informationen für den Benutzer aufbereitet, d.h.

- es wird eine Fehlerbeschreibung (Fehlerort, -art) angezeigt,
- es wird eine Sofortmaßnahme vorgeschlagen, die einer Eskalation der vorliegenden Situation entgegenwirkt,
- es werden potentielle Gegenmaßnahmen vorgeschlagen, die ergriffen werden können und bewertet werden sollen.

Aufgrund der Informationen kann der Anwender entscheiden, welche Aktionen durchgeführt werden.

Im folgenden wird davon ausgegangen, dass alle Optionen zur Weiterverarbeitung ausgewählt und aktiviert werden. Das Alarmmodul wird diese dann über einen weiteren Kanal an das Prognosemodul weiterleiten, welches die Koordination der simulationsbasierten Prognose durchführt.

8.1.5 Modul: Prognose

Das Prognosemodul erhält vom Alarmmodul

- Informationen zur vorliegenden Situation,
- eine Sofortmaßnahme und
- eine Liste von zu bewertenden Gegenmaßnahmen.

Zunächst wird das Prognosemodul eine vollständige Aktualisierung des Simulationsmodells durch die Datenquellen veranlassen. Anschließend wird die online-Kopplung kurzfristig angehalten, eine Kopie des aktuellen Simulationsmodells angelegt, ggf. eine Zustandstransformation aufgrund der Sofortmaßnahme durchgeführt und dieses Zustandsabbild des Modells gespeichert. Hierzu wird beim vorliegenden objektorientierten Simulationssystem das Simulationsmodell als Klasse aufgefasst (Urbild-Klasse - UK), die dupliziert wird (Abb. 98).

Mit der UK können anschließend temporär Objekte instanziiert werden, die jeweils die Bewertung einer Maßnahme repräsentieren. Entsprechend wird für jedes Element der Maßnahmenliste eine Modellinstanz von UK erzeugt. Durch geeigneten Programmcode aus der Datenbank oder des Modells wird diese Maßnahme aktiviert und die Simulation für einen definierten Bewertungszeitraum gestartet. Nach Simulationsende wird durch das Prognosemodul eine bereits im Ursprungsmodell zu erstellende Auswertungsmethode aufgerufen, die die zur Bewertung erforderlichen Simulationsergebnisse ermittelt und durch einen DISDACOS-Datenkanal zum nachfolgenden Bewertungsmodul sendet.

Analog zum Überwachungsmodul kann der rechenaufwendige Simulationsteil des Prognosemoduls hierbei ebenfalls verteilt erfolgen. Dazu wird zunächst UK generiert

und gespeichert. Da ein Simulationsmodell aus Architektursicht ebenfalls eine Datenquelle ist, kann auf UK als Attribut zugegriffen werden. Demgemäss kann das Prognosemodul mehrere Simulationssysteme mit UK starten, was hinsichtlich der modellmäßigen und zeitlichen Abgeschlossenheit der Rechenaufgabe, im Gegensatz zum großen Koordinierungsaufwand einer verteilten Simulation im allgemeinen Fall [151,152], keine Probleme bereitet. Der dennoch notwendige Synchronisationspunkt liegt dabei bei der Ergebnisübergabe zum Bewertungsmodul.

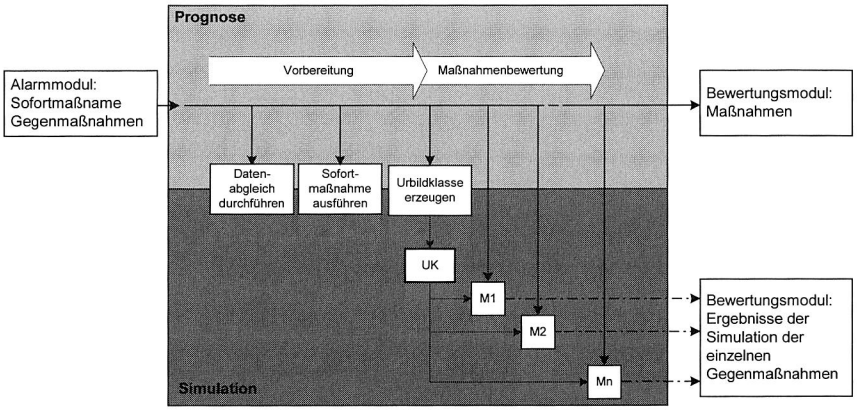


Abb. 98: Bewertung der potentiellen Maßnahmen: Vorbereitung des Simulationssystems, erzeugen der Urbildklasse und bewerten der Einzelmaßnahmen (M1 - Mn)

8.1.6 Modul: Bewertung und Auswahl der Maßnahme

Im Bewertungsmodul laufen die Ergebnisse aus den Simulationen des Prognosemoduls zusammen. Jede Einzelmaßnahme wird hier mit einer kurzen Beschreibung und ausgewählten Kennwerten der Prognosen und der bereits eingeleiteten Sofortmaßnahme angezeigt.

Liegen alle Teilergebnisse vor, erfolgt eine Auswertung der Simulationsergebnisse gemäß den Vorgaben des Anwenders, um die geeignetste Maßnahme zu finden (Abb. 99). Hierbei kann beispielsweise festgelegt werden, dass die beste Maßnahme bezüglich einer Minimum-/Maximumbewertung eines oder mehrerer zusammengefasster Kennwerte vorgeschlagen werden soll (maximale Ausbringung, minimaler Terminverzug usw.).

Es besteht entweder die Möglichkeit

- zunächst die Simulation aller Maßnahmen abzuwarten und dann eine Maßnahme individuell zu veranlassen,

- nach vollständiger Bewertung nach Verstreichen einer einstellbaren Zeit die beste Maßnahme automatisch zu ergreifen oder
- explizit eine Maßnahme zu selektieren, ohne dass weitere Ergebnisse abgewartet werden.

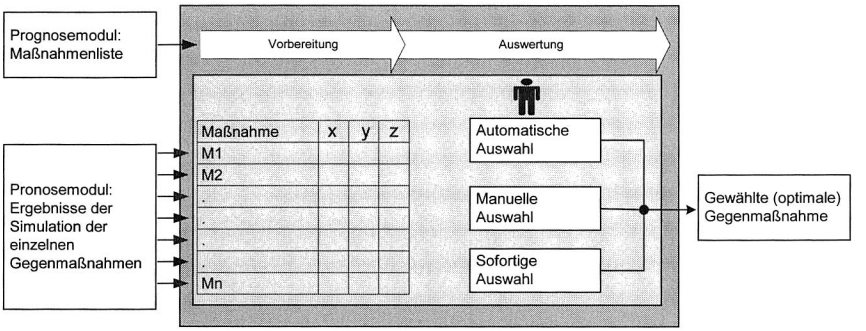


Abb. 99: Auswertung aller vom Prognosemodul berechneten Ergebnisse durch das Bewertungsmodul

Soweit dies möglich ist, wird die dann (manuell oder automatisch) ausgewählte Maßnahme über das Kopplungsmodul oder einen weiteren Wrapper aktiviert. Es ist beispielsweise denkbar

- einen Arbeitsauftrag einschließlich notwendiger Arbeitsanweisungen an den verantwortlichen Instandsetzer zu schicken,
- geeignete Maßnahmen im Leitstand zu ergreifen (Umdisposition von Aufträgen)
- in Produktionsnetzen eine weitere Störmeldung zu generieren, mit der beispielsweise ein Zulieferer über signifikant veränderte Zeit- und Mengenbedingungen innerhalb einer JIT (just in time) / JIS (just in sequence)-Kette unterrichtet wird. Mit einem eigenen Monitoring-Modul kann dieser dann analog geeignete Gegenmaßnahmen innerhalb seines Entscheidungshorizonts ergreifen.

8.2 Anwendungsszenario

Das im folgenden dargestellte Anwendungsszenario gibt eine Übersicht über alle in Kap. 7 und 8 entwickelten Module. Hierbei wird jedoch kein real bestehendes Produktionssystem in das Anwendungsszenario aufgenommen. Vielmehr sollen die konzeptionellen Stärken des Systems für typische Anwendungsfälle herausgearbeitet werden. Demgemäss wird auf die Anbindung eines realen Systems verzichtet und stattdessen das zu überwachende Produktionssystem in Form eines detailliert beobachtbaren Simulationsmodells nachgebildet.

Hierdurch ist gewährleistet, dass unterschiedliche Szenarien schnell dargestellt werden können, ohne dass hiervon ein produktives Realsystem betroffen wäre. Darüber hinaus wird so die Integration des Monitoringsystems auf der Basis von Fabrikplanungsergebnissen deutlicher herausgearbeitet: Die zielgerichtete Gestaltung der Datenkopplung und Überwachung erfolgt dabei zunächst aufgrund eines Simulationsmodells, welches während der Anlagenprojektierung entwickelt worden ist. In diesem werden anschließend die notwendigen Datenerfassungspunkte und Überwachungsmodule eingesetzt und hinsichtlich ihrer Zweckmäßigkeit, Angemessenheit und Vollständigkeit geprüft.

8.2.1 Struktur des betrachteten Produktionssystems

Entsprechend der vorangegangenen Überlegungen wurde ein Modell erstellt, welches ein einfaches, aber typisches Beispiel für den Einsatz der Simulation bei der Planung eines Produktionssystems ist. Dieses wurde gemäß den Vorüberlegungen schrittweise durch Strukturaufbau (Datenklasse S), Ablaufbeschreibung (Datenklasse A) und Parametrierung (Datenklasse P) realisiert.

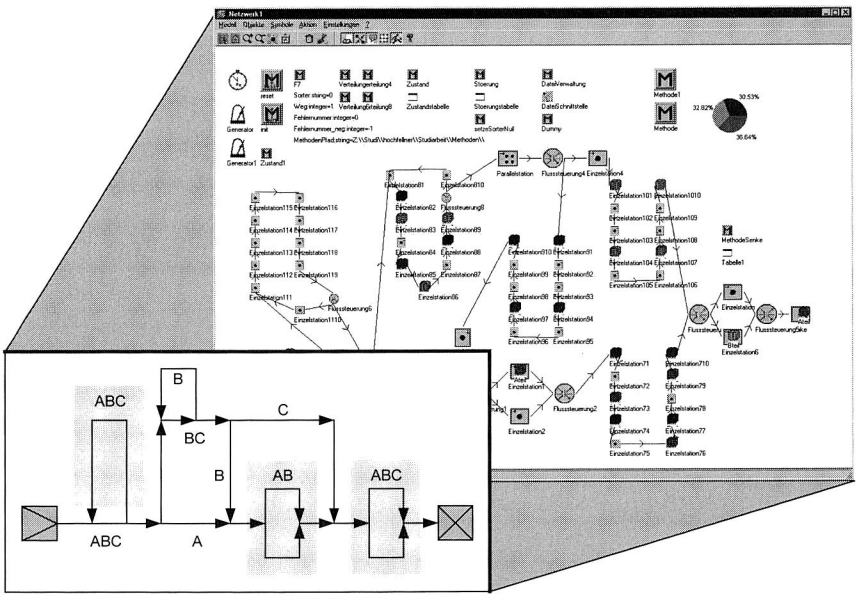


Abb. 100: Struktur des betrachteten Produktionssystems (Durchlauf der A-, B- und C-Teile) und Übersicht des daraus aufgebauten Simulationsmodells

Das Modell charakterisiert ein Montagesystem, in welchem drei unterschiedliche Produkte hergestellt werden können. Diese werden als A-,B- und C-Teile bezeichnet.

Sie gelangen über eine Quelle in das System und werden zunächst in einem Umlauflager zwischengespeichert. Der Fertigungsplan sieht für jedes Produkt teilweise unterschiedliche Arbeitsschritte vor. Im vorliegenden Beispiel wurden die Fertigungspläne so gewählt, dass durch die resultierenden Wege eine ausgeprägte Systemdynamik entsteht. Diese sind intuitiv schwer zu überblicken und der Einsatz der Simulation ist sinnfällig. Die Gesamtstruktur zeigt Abb. 100.

Gemäß dieser Struktur wurden dann die Abläufe beschrieben und entsprechende Parameter im Modell hinterlegt, wodurch das in Abb. 100 ebenfalls dargestellte Simulationsmodell aufgebaut wurde.

Vorbereitend auf das Monitoring werden weiterhin mehrere Wrapper erstellt, die den Zugriff auf die einzelnen Kenngrößen während des Simulationslaufs ermöglichen. Ferner enthält das Simulationsmodell einen Strukturscanner, der Strukturdaten in der Verwaltungsschicht registriert.

Abschließend werden in das Modell Störszenarien implementiert, welche durch den Benutzer aktiviert werden können. Diese werden im folgenden verwendet, um die Abläufe im Gesamtsystem zu verdeutlichen.

8.2.2 Struktur des Simulationsmodells zum Monitoring

Die Struktur des zum Monitoring eingesetzten Simulationsmodells orientiert sich an der Struktur des vorgehend beschriebenen 'realen' Produktionssystems (Abb. 101). Hierbei ist die abstrahierende Vorgehensweise der Modellbildung zu berücksichtigen, d.h. es sind Teile des Ursprungssystems zu vereinfachen (Strukturen), Einflussgrößen zu vernachlässigen (Abläufe) und Systemdaten (Parameter) mit einer der Datenbeschaffung immanenten Abweichung zu belegen.

Die Strukturbildung erfolgt gemäß der Zielstellung dieser Arbeit in zwei Schritten. Der aus Architektursicht interessante erste Schritt nutzt die Navigationskomponente der Verwaltungsschicht, um durch eine hierarchische Strukturübersicht die verfügbaren Datenquellen zu erhalten. Hieraus werden dann im Sinne der beschriebenen datengetriebenen Modellerstellung durch den Strukturgenerator die Grundzüge eines Simulationsmodells erzeugt.

Dieses wird anschließend im zweiten Schritt durch einen Simulationsexperten vervollständigend weitermodelliert. Das entstandene Modell wird dabei um weitere Datenkopplungsbausteine (z.B. zur Abfrage bereits verdichteter Informationen) und um Überwachungskomponenten ergänzt. Daraus entsteht sowohl der im Simulationssystem realisierte Monitorkern als auch die in Kap. 8.1.5 beschriebene Urbildklasse, welche zur Maßnahmenbewertung separat als Objekt zu speichern ist.

Der Datenabgleich erfolgt parallel zur Modellierung, indem statt der in einem Simulationssystem meist vorhandenen Standardbausteine die in Kap. 5 beschriebenen Datenkopplungsbausteine verwendet werden. Da diese funktional eine Erweiterung darstellen, können sie bei der Modellierung zunächst wie gewohnt genutzt werden. Werden anschließend die Datenkopplungsfunktionen benötigt, so kann der Anwender durch Angabe einer Datenquellenbezeichnung diese aktivieren. Die Nutzung erfolgt dann durch neue Attribute des Bausteins, die gemäß des Inhalts der zugrundeliegenden Datenquelle durch den Wrapper und das Kopplungsmodul des Simulators automatisch erzeugt werden.

Weiterhin kann der Simulationsexperte spezielle Prozeduren oder Methoden erstellen, die im Falle einer asynchronen Meldung direkt im Simulator aufgerufen werden. Dies ist im vorliegenden Beispiel dann notwendig, wenn die synchron zum Betriebsgeschehen laufende Simulation unterbrochen werden muss, um eine neue Urbildklasse zur Bewertung zu erzeugen.

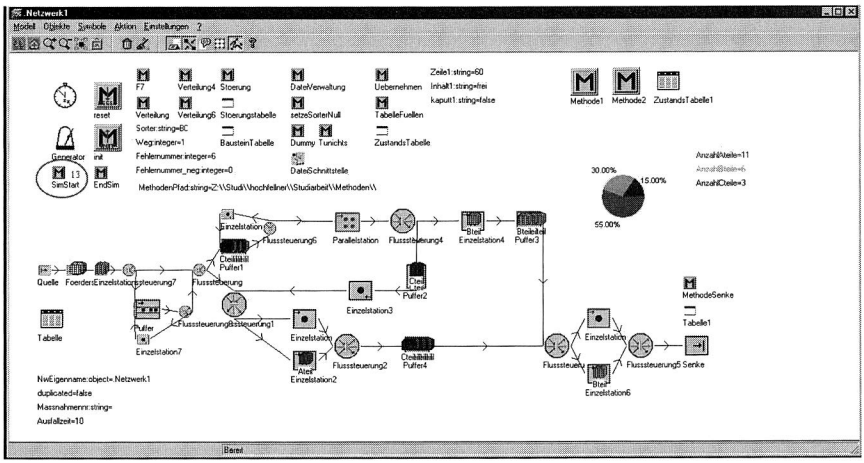


Abb. 101: Struktur des Produktionssystems im Simulationsmodell, welches zum Monitoring eingesetzt wird

Der synchrone Datenabgleich wird dadurch realisiert, dass die Simulation in 'Echtzeit' betrieben wird. Bei definierten Zeitmarken werden dann die Simulationsdaten für das Modell neu akquiriert. Diese Aktualisierungsrate ist so zu wählen, dass die Beobachtung und Bewertung des Betriebsgeschehens möglich ist. Im beschriebenen Szenario wird diese Rate dadurch bestimmt, dass zunächst die tatsächliche Zeit zur Aktualisierung einschließlich der Aufsetzzeit für die Kommunikation gemessen wird. Diese liegt bei hoher Netzwerklast, einer Systemlast auf DISDACOS und mehreren Datenquellenwrappern auf einem Rechner im vorliegenden Fall zwischen 2,5 und 4,5

Sekunden. Zusätzlich sind für die folgende Bewertung und Darstellung nochmals etwa 2 Sekunden zu veranschlagen. Insgesamt wurde so eine Aktualisierungsrate von 15 Sekunden gewählt, in die zu erwartete asynchrone Meldungen und Systemnebenzeiten (z.B. Garbage-Collection bei interpretierten Simulationssprachen) bereits mit einbezogen wurden.

8.2.3 Berücksichtigte Fehlerquellen und Gegenmaßnahmen

Während des Systementwurfs wurden bereits Störungen aufgelistet, die im späteren Betrieb potentiell auftreten können, beispielsweise Maschinenausfälle und Unterbrechungen des Materialflusses. Im vorliegenden Beispiel entsprechen die Störungen genau denen, die durch die Störszenarien definiert sind. Zur Demonstration ist die Störungen als MS-Excel Tabelle aufgebaut. Der Zugriff erfolgt über einen hierfür entwickelten Excel/DDE Wrapper. Dementsprechend ist es auch möglich, komplexere Fehlerkataloge in einer Datenbank abzulegen, weil aus Simulationssicht der Wrapper einen uniformen Zugriff auf diese Informationen erlaubt.

				Zu bewertende Gegenmaßnahmen					
0	**Bausteinname**	**Störungsbeschreibung**	**Sofort**	**MN1**	**MN2**	**MN3**	**MN4**	**MN5**	**NR**
1	.Netzwerk1.Foerderstrecke	Foederstrecke ausgefallen	ABC	F1MN1	F1MN2	F1MN3	F1MN4	F1MN5	1
2	.Netzwerk1.Flusssteuerung7	Keine / Folgefehler	0	none	none	none	none	none	0
3	.Netzwerk1.Einzelstation	Einzelstation ausgefallen	ABC	F3MN1	F3MN2	F3MN3	F3MN4	F3MN5	2
4	.Netzwerk1.Flusssteuerung	Keine / Folgefehler	0	none	none	none	none	none	0
5	.Netzwerk1.Flusssteuerung1	Keine / Folgefehler	0	none	none	none	none	none	0
6	.Netzwerk1.Einzelstation1	Einzelstation1 ausgefallen	AC	F6MN1	F6MN2	F6MN3	F6MN4	F6MN5	3
7	.Netzwerk1.Einzelstation2	Einzelstation2 ausgefallen	AC	F7MN1	F7MN2	F7MN3	F7MN4	F7MN5	4
8	.Netzwerk1.Flusssteuerung2	Keine / Folgefehler	0	none	none	none	none	none	0
9	.Netzwerk1.Parallelstation	Parallelstation ausgefallen	BC	F9MN1	F9MN2	F9MN3	F9MN4	F9MN5	5
10	.Netzwerk1.Flusssteuerung1	Keine / Folgefehler	0	none	none	none	none	none	0
11	.Netzwerk1.Einzelstation3	Einzelstation3 ausgefallen	C	F11MN1	F11MN2	F11MN3	F11MN4	F11MN5	6
12	.Netzwerk1.Einzelstation4	Einzelstation4 ausgefallen	B	F12MN1	F12MN2	F12MN3	F12MN4	F12MN5	7
13	.Netzwerk1.Flusssteuerung3	Keine / Folgefehler	0	none	none	none	none	none	0
14	.Netzwerk1.Einzelstation5	Einzelstation5 ausgefallen	0	F14MN1	F14MN2	F14MN3	F14MN4	F14MN5	8
15	.Netzwerk1.Einzelstation6	Einzelstation6 ausgefallen	0	F15MN1	F15MN2	F15MN3	F15MN4	F15MN5	9
16	.Netzwerk1.Flusssteuerung5	Keine / Folgefehler	0	none	none	none	none	none	0
17	.Netzwerk1.Senke	Senke ausgefallen	ABC	F16MN1	F16MN2	F16MN3	F16MN4	F16MN5	10

Tabelle 7: Im Prototypen potentiell auftretende Störungen mit Bezeichnung von Fehlerort, -art, einzuleitenden Sofortmaßnahmen und zu bewertenden Gegenmaßnahmen

Diese in Tabelle 7 aufgeführten Meldungen enthalten die in Kap. 8.1.3 beschriebenen Informationen. Mit ersten Simulationsläufen im konkreten Simulationsmodell konnten zusätzlich auch einige mögliche Folgefehler identifiziert werden, die in den Katalog der 'berücksichtigen Fehlerquellen' aufgenommen wurden.

Die aufgeführten Gegenmaßnahmen (hier: MN1 bis MN5) verweisen für jede Aktion auf eine im Monitor anzustoßende Simulationsmethode. Beispielsweise verweist F1MN2 auf Fehler Nr. 1, zu dessen Behebung Maßnahme Nr. 2 angewendet werden soll. Alternativ kann im Falle eines Simulationssystems mit interpretierter Ablaufbeschreibungssprache die Aktionsbezeichnung als Referenz auf den Code einer Methode/Prozedur/Funktion sein, die geladen und ausgeführt wird. Im vorliegenden Fall wurden die Gegenmaßnahmen fest in den Monitor kodiert.

8.2.4 Erstellung und Nutzung anwendungsspezifischer Kontrollbausteine

Wie in Kap. 8.1 diskutiert, wird das Monitoringsystem durch die Module

- Laufzeitüberwachung, Alarm und Methodenausführung bedarfsweise aktiviert und
- Datenkopplung, Laufzeitsteuerung und Bewertung gesteuert.

Die Aktivierungsmodule werden im Simulationssystem zum Monitoring (Kap. 8.2.2) und unter Verwendung der Überwachungsbausteine aus Kap. 7 realisiert.

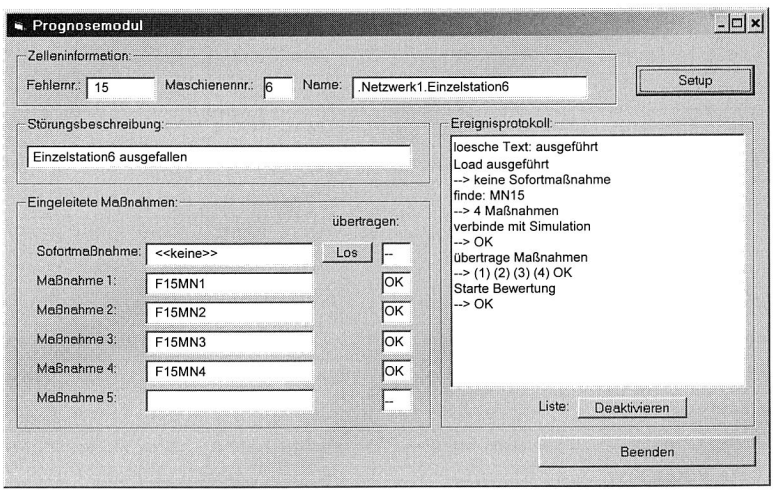


Abb. 102: Darstellung des Prognosemoduls, mit dem im Fehlerfalle die Laufzeitsteuerung des Monitoringsystems erfolgt

Die Kontrollmodule zur Laufzeitsteuerung und Bewertung werden als eigenständige Anwendungen entworfen, welche gemäß den Anforderungen die Informationsverar-

beitung und -steuerung übernehmen. Die Datenkopplung wurde im Simulationssystem realisiert.

Das Prognosemodul mit Laufzeitsteuerung wird Abb. 102 dargestellt.

Hier dem Benutzer Informationen angezeigt, die ausgehend von einer Fehlernummer aus der Fehlerdatenbank ermittelt werden. Insbesondere wird an dieser Stelle auch der zu bewertende Maßnahmenkatalog aufgelistet.

Gemäß des Konzepts werden diese Daten sofort an das Bewertungsmodul und das Simulationsmodell zur Berechnung weitergeleitet. Das Simulationsmodell selbst nimmt dann die Erstellung von UK und die Ermittlung der notwendigen Kennwerte vor.

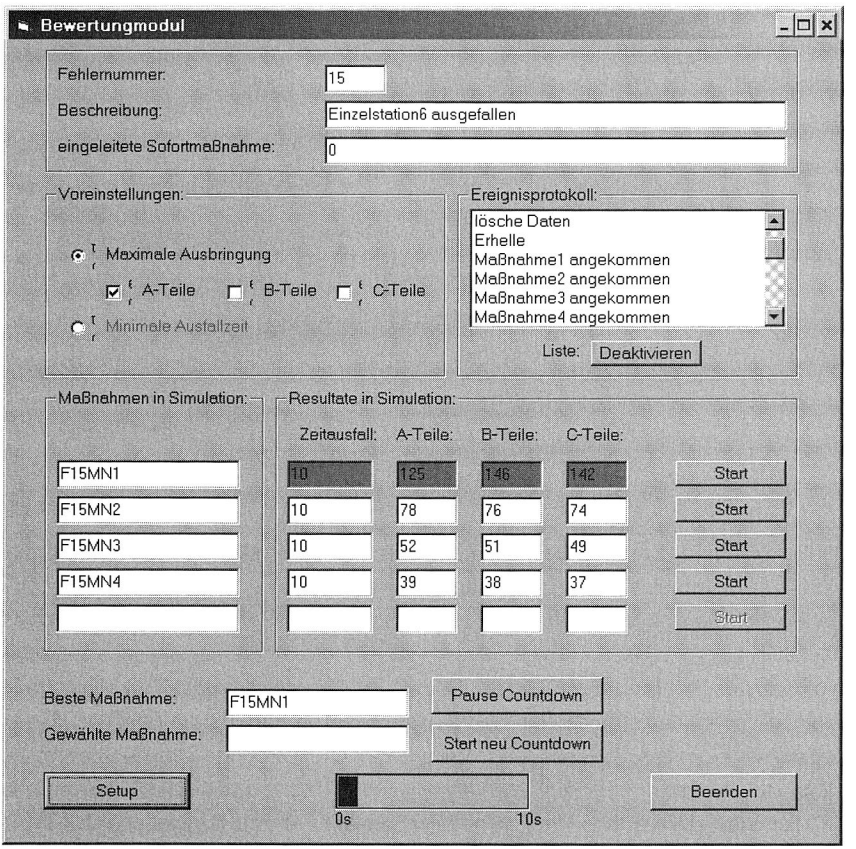


Abb. 103: Benutzeroberfläche des Bewertungsmoduls zur Darstellung der Simulationsergebnisse

Die berechneten Werte (in diesem Fall Zeitausfall, Ausbringung an A-,B- und C-Teilen) werden sofort nach Beendigung des jeweiligen Simulationslaufs an das Bewertungsmodul (Abb. 103) weitergemeldet. Hier kann der Nutzer zunächst die Ausgangsdaten der Simulationsläufe sehen (Fehlernummer, Beschreibung usw.) und zusätzlich das Auswahlkriterium für die beste Gegenmaßnahme wählen.

Sobald mindestens eine bewertete Maßnahme eingetroffen ist, kann der Benutzer diese übernehmen. Andernfalls wartet das Bewertungsmodul die Beendigung der gesamten Maßnahmenbewertung ab und schlägt aufgrund der Voreinstellung eigenständig eine Gegenmaßnahme vor.

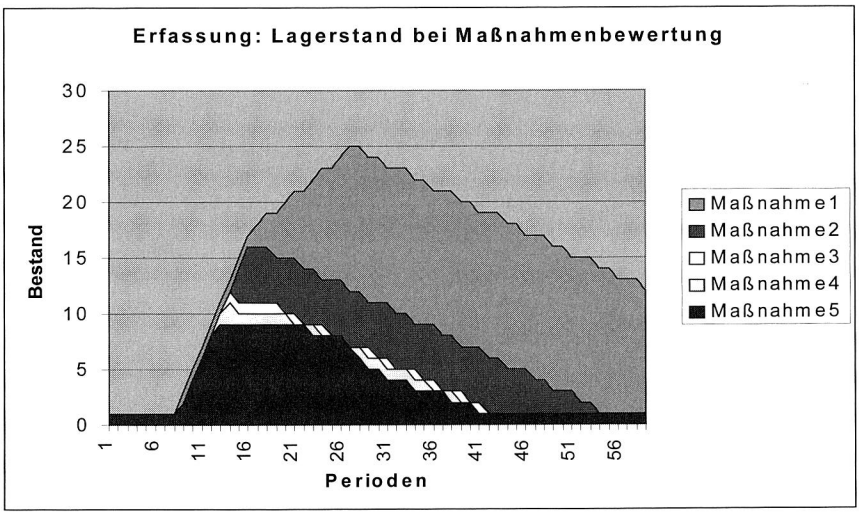


Abb. 104: Gegenüberstellung verschiedener Gegenmaßnahmen

Danach kann der Benutzer beginnen, die entsprechende Maßnahme und geeignete Aktivitäten umzusetzen. Zusätzlich ist es aber auch möglich, neben den stark abstrahierten Kennwerten der Bewertung sich detailliertere Informationen zum Simulationslauf anzeigen zu lassen. Beispielweise ist es möglich, das dynamische Verhalten des Systems während der Störperiode zu visualisieren, bzw. die jeweiligen Maßnahmen nochmals gegeneinander zu stellen, wie Abb. 104 verdeutlicht: An diesem Beispiel ist zu erkennen, dass bei Beginn der neunten Periode eine Störung aufgetreten ist. Die Bewertung zeigt, dass in diesem Fall Maßnahme '5' am geeignetsten ist. Hingegen erweist sich Maßnahme '1' in diesem Fall aus Simulationssicht als so ungeeignet, dass innerhalb des betrachteten Zeitraums der Normalbetrieb nicht wieder aufgenommen werden konnte.

Das Prognosemodul meldet zuletzt die Abarbeitung des Auftrages und gibt damit die gesamte Meldekette (Alarmmodul, Prognose, Simulation, Bewertung) für weitere Aufträge frei.

8.3 Diskussion und Bewertung der Ergebnisse

Im vorliegenden Kapitel wurden die Ergebnisse der Kap. 6 (Datenkopplung) und 7 (Überwachung) zu einem simulationsbasierten online-Monitoring zusammengefasst und an einem Prototypen demonstriert. Die Verarbeitung der Informationen im Prototypen erfolgte auf der Grundlage von DISDACOS. Um den Informationsfluss zielbezogen steuern zu können, wurden weiterhin die Module Datenkopplung, Laufzeitsteuerung und Bewertung konzipiert und realisiert. Es konnte gezeigt werden, dass durch den Einsatz von DISDACOS mehrere eigenständige Werkzeuge eine neue funktionale Gesamtheit darstellen. DISDACOS übernimmt dabei die Aufgabe einer Middleware für Simulationszwecke. Die Gesamtheit entspricht demzufolge einem Framework für ein simulationsbasiertes Online-Monitoring.

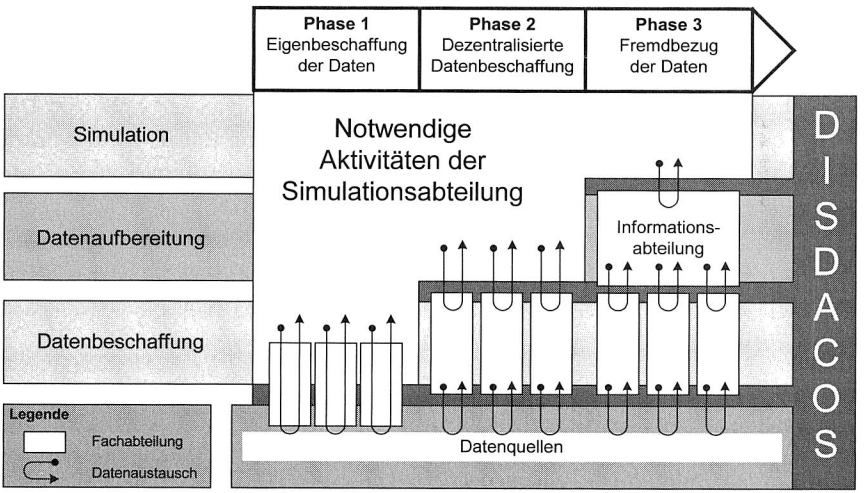


Abb. 105: Datenerfassung und Aufbereitung als Dienstleistung

Zur Bewertung des Gesamtkonzepts ist es zuletzt notwendig, die Integrierbarkeit eines solchen Systems in die Planungs- und Betriebsabläufe zu untersuchen. Die hier angestellten Überlegungen greifen dazu die Sichtweise auf, Informationen und Daten als Wirtschaftsgut anzusehen, welche unter Ressourceneinsatz zu beschaffen, zu verarbeiten und regelmäßig zu pflegen sind. Es soll daher der Frage nachgegangen werden, inwieweit die Beschaffung des "Produktes" Daten durch eine strategisch zu

entscheidende "make-or-buy" (Eigenfertigung oder Fremdbezug) Vorgehensweise organisiert werden kann und welche Unterstützung DISDACOS dabei bietet.

Zu diesem Zweck wird im folgenden eine dreistufige Vorgehensweise vorgestellt, welche ausgehend von der gängigen Praxis der Eigenbeschaffung von Daten durch den Simulationsexperten über einen Zwischenschritt letztlich die Datenbereitstellung an eine dedizierte Instanz ("Informationsabteilung") überträgt (Abb. 105).

8.3.1 Eigenverantwortliche Datenbeschaffung (Phase 1)

Bei der eigenverantwortlichen Datenbeschaffung wird der Simulationsexperte gemäß den Zielvorgaben der Studie das Modell bilden. Entsprechend müssen Daten beschafft werden. Zu diesem Zweck wird von Seiten der Simulationsabteilung oft direkt auf die Daten- und Informationsquellen der Planung oder Produktion zugegriffen, um daraus die benötigten Informationen und Parameter zu extrahieren.

Diese Vorgehensweise wird immer dann mit einem hohen Aufwand verbunden sein, wenn Informationen in heterogenen Netzen eines Werksverbunds zu lokalisieren und abzufragen sind. Die dabei zu überwindenden Schwierigkeiten sind meist technische wie organisatorische Probleme des Zugriffs auf Datenbestände. Unterschiedliche Datenbanksysteme für verschiedene Informationen sind für die Aufgabe einer umfassenden Datenrecherche ebenso hinderlich wie Zugriffsbeschränkungen von Datenbanken auf Grund von Betriebsgeheimnissen und "Kompetenzen". Auch mangelnde Informationen zur Topologie der Rechnerstrukturen im Betrieb können ein Hindernis darstellen. Der entscheidende Nachteil dieser Vorgehensweise liegt jedoch im Aufwand, welcher der Simulationsabteilung entsteht. Es muss immer wieder Zeit und Personal zur Datenbeschaffung aufgewendet werden, wodurch für die Kernkompetenzen (Modellbildung, Experimente, Ergebnisauswertung) weniger Ressourcen zur Verfügung stehen.

Zur Verdeutlichung des Nutzens von DISDACOS in dieser Phase sind zwei Fälle zu unterscheiden:

Im ersten Fall muss davon ausgegangen werden, dass DISDACOS zwar installiert ist, jedoch noch keine Informationen durch das System verfügbar sind. In diesem Fall werden zunächst die vorhandenen Wrapper verwendet, um gängige Datenquellen zu nutzen. Dies wären die DDE- und Datenbank-Wrapper. Mit diesen werden dann entweder bestehende Datenverarbeitungssysteme für die Simulation angekoppelt oder es wird eine neue Datenbank erstellt, in die alle für die erste Simulationsstudie notwendigen Informationen hinterlegt werden. Anschließend kann dann überlegt werden, welche weiteren Informationsquellen anzubinden sind. Für diese werden dann

individuelle Wrapper erstellt, womit das Gesamtsystem entsprechend der regelmäßig anstehenden Aufgaben sukzessive ausgebaut wird.

Im zweiten Fall sind Informationen durch das System verfügbar und DISDACOS kann bereits bei der Modellerstellung eingesetzt werden. Dazu kann aus den vorhandenen Informationsbeständen der notwendige Ausschnitt ausgewählt werden. Mittels des Modellgenerators werden anschließend die Grundzüge des Simulationsmodells weitgehend automatisch erstellt. Im folgenden Schritt der genaueren (manuellen) Nachmodellierung durch den Experten werden dann die Datenbestände bedarfsweise (analog zum ersten Fall) erweitert.

Durch eine regelmäßige Nutzung entwickelt sich so ein Netzwerk von Datenquellen, die für die regelmäßig anfallenden Aufgaben und typischen Simulationsgegenstände bei Modellierung und Parametrierung zur Verfügung stehen.

8.3.2 Dezentralisierte Datenbeschaffung (Phase 2)

Ausgehend von einem in der Simulationsabteilung installierten DISDACOS-System kann in einer weiteren Phase begonnen werden, die Datenbeschaffung zu dezentralisieren. Dieser Schritt ist durch das Ziel motiviert, die Simulation betriebsbegleitend einzusetzen, d.h. Daten direkt aus dem Betriebsgeschehen zu erlangen. Deshalb bietet es sich an, Datenabgriffe und Aufbereitung nicht ausschließlich in der Simulationsabteilung abzuwickeln, sondern vielmehr an der Stelle, an welcher sich die Daten befinden, d.h. in den produzierenden Bereichen des Unternehmens/Werks. Hierdurch lässt sich in geeigneter Weise das lokale Know-how zu den Strukturen der Informationsbestände nutzen. Durch die kooperative Einbindung der zuständigen Fachabteilungen kann wichtiges Datenmaterial einfacher identifiziert, aufbereitet und systematisch bereitgestellt werden.

Der Beitrag der Fachabteilung erstreckt sich jedoch nicht nur auf die Nennung von Datenquellen, sondern ist ausschlaggebend für die Datenqualität. Gemeinsam mit der Fachabteilung kann nicht nur festgelegt werden, welche Daten einen realen Prozess beschreiben, sondern es wird zudem klarer, welche messbaren Größen eines Produktionsprozesses als Basis für die Kennwerte des Prozesses dienen können und wie ggf. fehlerhafte Daten korrigiert werden müssen.

Ebenso kann die Fachabteilung wichtige Hinweise darauf geben, ob die verwendeten Größen u.U. von nicht in der Simulation erfassten Nebenbedingungen beeinflusst werden. Dieses Prinzip der Separabilität soll am Beispiel einer Drehmaschine verdeutlicht werden: Die Standzeit eines Drehmeißels ist nur zusammen mit der verwendeten Schnittgeschwindigkeit aussagekräftig. Die alleinige Abbildung der Standzeit im Modell wäre nicht sinnvoll.

Organisatorisch lassen sich zusätzlich auch eventuell vorhandene Zugangsbeschränkungen umgehen, da die Fachabteilung die Kontrolle über die bereitgestellten Daten behält. Dieser Punkt ist in der Praxis häufig entscheidend, weil erst durch eine Betonung der Kompetenz der jeweiligen Fachabteilung die Grundlage für eine für beide Seiten zufriedenstellende Zusammenarbeit geschaffen wird.

Durch den Einsatz von DISDACOS wird dabei erreicht, dass die u.a. in der Fachabteilung aufbereiteten Informationen auch in einer einheitlichen Form genutzt werden. Die Simulationsabteilung nutzt somit das lokale Fachwissen der betroffenen Unternehmensbereiche. Das Konzept der verteilten Datenhaltung bei DISDACOS erschließt deshalb eine arbeitsteilig-kooperative Datenbeschaffung, ohne dass bei der anschließenden Nutzung dieser Informationen für den Simulationsexperten ein Zusatzaufwand entstehen würde.

Technisch wird bei der Realisierung dieser Phase der gleiche Ansatz verfolgt, der bereits in Phase 1 vorgeschlagen wurde: Zunächst werden Daten erschlossen, die mit vorhandenen Wrappern verfügbar gemacht werden können. Im folgenden werden neue Wrapper erstellt, die weiteres Datenmaterial an DISDACOS anbinden, wozu die dann bereits gewonnen Erfahrungen der Simulationsabteilung aus Phase 1 einen wichtigen Beitrag leisten werden.

8.3.3 Fremdvergabe der Datenbeschaffung (Phase 3)

Die Verlagerung der Datenquellenpflege von der Simulations- in Fachabteilungen kann aus operativer Sicht als Ideal angesehen werden, weil eine optimale Verteilung von Verantwortungen vorherrscht. In der Praxis dürfte jedoch ein merkliches Hemmnis darin bestehen, dass die Fachabteilungen nicht die notwendigen Ressourcen bereitstellen wollen/können, um dieses System langfristig erfolgreich aufrechterhalten zu können.

Sollte also ein derartiger Ansatz zur lokalen Datenaufbereitung weiter verfolgt werden, ist es notwendig, eine weitere Instanz einzusetzen, welche von der Simulationsabteilung angerufen wird, sobald der Bedarf nach Daten besteht. Dieser Instanz müssen, um die angesprochenen Aufgaben lösen zu können, eine Reihe von Kompetenzen eingeräumt werden. So muss sie die Befugnis besitzen, in den Fachabteilungen einheitliche Schnittstellen zum Datenabgriff durch die Simulationsabteilung zu installieren. Des weiteren hat sie dafür Sorge zu tragen, die verschiedenen Informationen zur Datenaufbereitung aus Simulationsabteilung und lokalen Fachabteilungen zu sammeln. Ebenso sind die Fachabteilungen durch die Unternehmensführung zu ermutigen, Ihr Wissen zu den Datenstrukturen der Werke bei einem Zugriff auf die benötigten Datenquellen aktiv einzubringen. Sich ergebende neue Erfahrungen während der Auswertungen sind gleichermaßen an die zentrale Informationsabteilung

weiterzuleiten. Aus Sicht der Simulationsabteilung findet somit eine Fremdvergabe der Datenbeschaffung an die Informationsabteilung statt. Die Daten und Informationen werden dementsprechend 'eingekauft'.

Bei der Umsetzung von Phase 3 sind zwei Gesichtspunkte zu berücksichtigen.

Erstens ist bei der Einrichtung einer Informationsabteilung deren Rolle im Unternehmen zu klären. Dabei kann zunächst angenommen werden, dass die Informationsabteilung im Sinne eines profit-centers operieren soll. Das Hindernis sind dabei jedoch ganz klar mangelnde Befugnisse, da seitens der Informationsabteilung Datenanforderungen kaum durchgesetzt werden können. Erfahrungsgemäß wird davon vor allem die Zusammenarbeit mit Abteilungen betroffen sein, deren Position (Autonomie) durch eine höhere Informationstransparenz gefährdet sein könnte.

Aus diesem Grund ist die strategische Entscheidung zur Einrichtung einer Informationsabteilung geeignet zu unterstreichen. Am ehesten geeignet wäre dabei, der Informationsabteilung den Status einer internen Revision beizumessen, der gegenüber alle Unternehmensteile in der Sache auskunftspflichtig sind. Aufgrund dieser weitreichenden Befugnisse muss dann jedoch darüber nachgedacht werden, welche Dienste eine solche Abteilung neben der Simulationsabteilung anderen Teilen des Unternehmens anbieten kann, um Leerkosten dieser Abteilung zu vermeiden und gleichzeitig die Akzeptanz für deren Arbeit zu steigern.

Zweitens ist technisch der Leistungsaustausch zwischen Simulations-, Informations- und Fachabteilung zu klären. Die Lösung wird aus Sicht der Simulationsabteilung und deren Zugriff auf alle notwendigen Information durch DISDACOS geliefert: Der Verwaltungskern von DISDACOS kann ebenso verteilt aufgebaut werden wie jede andere Menge von Datenquellen. Somit kann die zentrale Verwaltung von der Simulations- in die Informationsabteilung verlagert werden, ohne dass hierzu die Bestandteile der Architektur verändert werden müssen. Informationstechnisch werden letztlich zwei Datendomänen logisch miteinander verbunden, wie in Kap. 6.7.1 beschrieben wurde.

Im Rahmen der Diskussion über die Sicherheit der Architektur (Kap. 4.2.1) kann DISDACOS dann auch zur Leistungsabrechnung eingesetzt werden, da die Zugriffsrechte auf einzelne Datenquellen beispielsweise an die Zahlung eines vereinbarten Preises geknüpft werden können. Dies führt dann letztlich zu einem Lizenzierungs- und Abrechnungsdienst, der die Nutzungsdauer von Datenquellen messen und entsprechend abrechnen kann. Dieser sollte dann nicht nur von der Informationsabteilung, sondern in diesem Szenario auch von der Simulationsabteilung genutzt werden, um beispielsweise die Abrechnung von Simulationsergebnissen zu ermöglichen.

8.3.4 Technische und wirtschaftliche Grenzen des Einführungsprozesses

Für die Simulationsabteilung hat der Einsatz von DISDACOS den Vorteil einer Entlastung von nicht wertschöpfenden Aktivitäten. Die Durchführungsdauer einer Simulationsstudie kann hiermit verkürzt werden. Dabei ist trotzdem gewährleistet, dass bei der Simulations-/Informationsabteilung als zentraler Stelle das Wissen zur Aufbereitung der Daten gesammelt wird und in die Verarbeitung der Daten einfließt.

Die make-or-buy Entscheidung kann für einzelne Daten fallweise getroffen werden. Die Entscheidungsgrundlage ist dabei nun aber nicht nur vom rein technisch erforderlichen Datenbedarf abhängig, sondern kann vielmehr anhand der Kosten und des Wertes der anzufordernden Daten getroffen werden: Für die Simulationsabteilung wird deshalb eine Entscheidung aufgrund der Frage getroffen, ob der Wert der Information die Kosten zur Datenbeschaffung übersteigt. Weil in diesem Fall idealerweise auch keine kalkulatorischen, sondern tatsächlich zahlungswirksame Kosten zur Bewertung vorliegen (Kostentransparenz), ist zu erwarten, dass die Kosten zur Durchführung einer Simulationsstudie insgesamt sinken, da vor allem die Datenbeschaffung 'sparsam' durchgeführt wird.

Dies soll an einem Beispiel verdeutlicht werden: Ein Simulationsexperte hat einen Stundensatz von 150 €. Demgegenüber sind für den Sachbearbeiter der Informationsabteilung 90 € je Stunde zu berechnen. Zur Beschaffung eines minutengenauen Arbeitszeitmodells auf der Basis von Betriebsdaten wird ein Zeitaufwand von 50 Stunden erwartet, d.h. es entstünden Kosten von 7500 € bzw. 4500 € (es wird die gleiche Datenqualität angenommen). Die Entscheidung, diese Daten von der Informationsabteilung zu beziehen, ist in diesem Falle deutlich vorteilhafter, wobei hier die zu erwartende schnellere Bearbeitung durch den Sachbearbeiter aufgrund von Erfahrung, schnellerem Datenzugriff oder besseren Werkzeugen nicht berücksichtigt wird. Die entstehenden Kosten von 4500 € wären für die Simulationsabteilung akzeptabel, da der zu erwartende Wert (Simulationsergebnisse) bei weitem höher liegt. Dies gilt sogar dann, wenn die Informationsabteilung diesen Auftrag wegen Auslastung erst später bearbeiten will und die Simulationsabteilung einen höheren Betrag zu zahlen hätte, um diese Informationen termingerecht zu erhalten. Durch das Konzept der verteilten Datenquellen könnten mit DISDACOS dann ab Kosten von 7500 € die Datenquellen wieder in die Simulationsabteilung zurückverlagert werden (make statt buy), ohne dass das eigentliche Simulationsmodell dafür verändert werden müsste. Für den Kostenstellenverantwortlichen wäre in diesem Fall die hierdurch gewonnene Transparenz nützlich, um das Aufwand-/Nutzenverhältnis der Simulationsstudie zu überdenken, d.h. entweder vom Auftraggeber der Simulation selbst einen höheren Preis zu verlangen, die Qualität (Zeit, Kosten und/oder Funktion) der Simulationsstu-

die zu reduzieren, einen alternativen - günstigeren- Informationsbeschaffer zu suchen oder letztlich die Simulationsstudie wegen Unwirtschaftlichkeit abzulehnen.

Somit wird durch diese wirtschaftlich motivierten Effekte auch die Bewertung der Simulationswürdigkeit einer Studie objektiviert. Jede positive Entscheidung zum Einsatz der Simulation wird im Bereich der Datenbeschaffung durch DISDACOS flexibel und offen unterstützt.

9 Zusammenfassung und Ausblick

Die fortschreitende Entwicklung der Rechnertechnik und der Softwaresysteme hat auch wichtige Voraussetzungen für eine sinnvolle, breite Anwendung der Simulationstechnik geschaffen. Das Anwendungsspektrum moderner Simulationswerkzeuge kann mittlerweile alle wichtigen Bereiche der Produktionstechnik abdecken. Aufgrund der Komplexität der Abläufe und Prozesse auf einem hohen, analytisch nicht mehr erfassbaren Niveau ist die Simulation zum unverzichtbaren Werkzeug geworden, um trotz enger werdendem Zeithorizont eine hohe Planungsqualität zu gewährleisten. Neben der Komplexitätsbeherrschung liegt die Attraktivität der Simulation darin begründet, dass Simulationsmodelle problembezogene Systembeschreibungen sind und damit als "neutrale Projektsflächen" für verschiedene Modellwelten (Fachsichten) dienen. Folgerichtig wird dieses Werkzeug als potentielle Plattform für die interdisziplinäre und durchgängige Kommunikation im Rahmen eines *concurrent engineerings* gesehen.

In der Praxis ist diese Durchgängigkeit erst punktuell erreicht. Dies zeigt sich vor allem bei der Beschaffung simulationsrelevanter Daten, für deren Beschaffung - trotz zunehmender Investitionen in Informationstechnologien - mittlerweile ein Aufwand von bis zu 50% an dem Gesamtaufwand einer Simulationsstudie angenommen werden kann. Bei der genaueren Untersuchung dieses Anwendungshemmnisses wurde deutlich, dass die Erstellung von Simulationsmodellen durch die Heterogenität der verwendeten Informationstechnologien und Beschreibungsformen behindert wird. Gleichzeitig konnte gezeigt werden, das simulationsrelevante Daten in hochautomatisierten Produktionssystemen durchaus in ausreichender Menge und Qualität verfügbar sind. Um diese simulationsseitig zu erreichen, wurde ein System konzipiert, welches solche Informationen als Primärdaten für Simulationszwecke verfügbar macht. Ausgangsbasis waren Informationen, welche nach dem Stand der Technik zur Beschreibung von Simulationsmodellen, -experimenten und -ergebnissen erforderlich sind. Diese wurden anschließend in generischen Datenklassen zusammengefasst und als Implementierungsgrundlage für eine objektorientierte Implementierung auf Basis der standardisierten Middleware CORBA verwendet. Durch die Nutzung dieser Middleware war es möglich, die entwickelte Architektur nach dem Prinzip der Verkapselung, unabhängig von der Lokalität von Objekten (Ortstransparenz) und von spezifischen Betriebssystemen oder Programmiersprachen (Plattforminvarianz) aufzubauen und betreiben zu können. Die Umsetzung des System erfolgte auf UNIX und WINDOWS Betriebssystemen, wobei fallweise die Programmiersprachen C++ und JAVA zum Einsatz kamen.

Mit diesem System wurde es möglich, auf simulationsrelevante Informationen in einer einheitlichen Weise zuzugreifen. Dies wurde auf Seite der Datenquellen (z.B.

Anlagenteile, Maschinen, Datenbanken usw.) durch Wrapper erreicht, die die sehr unterschiedlichen datenquelleneigenen Formate in einheitliche Architekturformate umsetzen und von einem für diese Architektur erweiterten Simulationssystem abgegriffen werden können. Die Kopplung zwischen Anlagendaten und Simulation erfolgt in einer Verwaltungsschicht. Durch sie wird der simulationsseitige Zugriff auf die bereitgestellten Informationen abgewickelt und die Konsistenz der Daten überwacht. Darüber hinaus kann der Anwender dieses Systems die hier vorhandenen Informationsbestände anhand von verschiedenen Hierarchisierungskriterien sichten und Datenquellen gezielt auswählen. Die jeweilige Auswahl kann anschließend als Menge von Strukturelementen in ein Simulationsmodell übernommen werden. Das ausgewählte Hierarchisierungskriterium wird dabei korrekt als Modellhierarchie übertragen (datengetriebene Modellierung). Bei der Initialisierung eines Simulationsexperiments können die Informationen der ursprünglichen Datenquellen dann direkt abgegriffen werden, wodurch ein aktueller Bezug zwischen Anlagendaten und Modellparametern möglich wird. Der gesamte Zugriff einschließlich der Verwaltung der Datenbestände erfolgt in der gewohnten Begriffswelt des Simulationsanwenders.

Die Leistungsfähigkeit dieses verteilten Datenbeschaffungssystems (*distributed data collecion system* - DISDACOS) hat anschließend dazu motiviert, das Simulationsmodell als Monitor einzusetzen. Dabei werden die aktuell verfügbaren Daten aus der Anlage in einem Simulationsmodell visualisiert und permanent auf signifikante Abweichungen (Störungen) überwacht. Für den Fall eines Störereignisses kann die Simulation dann genutzt werden, um ausgehend von der aktuellen Situation eine Prognose über die Auswirkungen dieser Störung zu erstellen, bedarfsweise mögliche Entstörmaßnahmen automatisch zu bewerten und dem Anwender vorzuschlagen. Das gesamte System (DISDACOS, Monitor und Prognosemodul) wurde für eine exemplarischen Aufgabenstellung aufgebaut und damit der Anwendernutzen nachgewiesen. Abschließend wurde eine schrittweise Integration des entwickelten Systems unter technischen und wirtschaftlichen Gesichtpunkten diskutiert. Die Potentiale bei der Umsetzung einer strategischen Entscheidung zur Simulation in einem Unternehmen wurden hierbei nochmals verdeutlicht.

Werden diese Überlegungen konsequent weitergeführt, so sollte der Einsatz dieser Architektur zukünftig nicht alleine auf die Simulation als 'Datenverbraucher' beschränkt bleiben, sondern gezielt den Durchgängigkeitsgedanken vorantreiben. Es werden Systeme entstehen, die sich vollständig in das Planungs- und Engineeringumfeld einbetten lassen und die Abwicklungsschritte des Anlagenlebenszyklusses umfassend unterstützen. Das simulationsgestützte Erweitern von Wissensbasen für Diagnosezwecke, die Nutzung der Simulation während der Inbetriebnahme und die Gestaltung von autonom operierenden, lose gekoppelten Programmsystemen (Agenten) stellen interessante Ergänzungen dieser Arbeit dar.

Glossar

API	Application Programming Interface
BDE	Betriebsdatenerfassung
BDSG	Bundesdatenschutzgesetz
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAM	Computer Aided Manufacturing
CAP	Computer Aided Planning
CLSID	Class Identifier
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
DB	Datenbank
DBMS	Datenbankmanagementsystem
DCOM	Distributed --> COM
DDE	Dynamic Data Exchange
DISDACOS	Distributed Data Collection System
DLL	Dynamic Link Library
DMSO	Defende Modeling and Simulation Office
DoD	Department of Defense
FOM	Federation Object Model
GEM	Generic Equipment Model
GUID	Global Unique Identifier
HLA	High Level Architecture
ICAM	Integrated Computer Aided Manufacturing
IDEF-0	ICAM Definition
IDL	Interface Definition Language
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IID	Interface Identifier
IP	Internet Protocol
JDBC	Java Database Connectivity
MCBA	Mean Cycles Between Assists
MCBI	Mean Cycles Between Failures
MTBA	Mean Time Between Assists

MTBF	Mean Time Between Failures
MTBI	Mean Time Between Interrupts
MTOL	Mean Time Off Line
MTTA	Mean Time To Assist
MTTR	Mean Time To Repair
OLE	Object Linking and Embedding
OMA	Object Management Architecture
OMG	Object Management Group
OMT	Object Model Template
OPC	--> OLE for Process Control
ORB	Object Request Broker
OSF	Open Software Foundation
OSI	Open Systems Interconnection
PPS	Produktionsplanung und -steuerung
RAM	Relialility, Availablity, Maintainability
RMI	Remote Method Invocation
RPC	Remote Procedure Call
RTI	Runtime Infrastructre
SECS	Semiconductor Equipment Communication Standard
SOM	Simulation Object Model
SQL	Structured Query Language
SSL	Secure Socket Layer
STEP	Standard for the Exchange of Product Model Data
TCP	Transmission Control Protocol
TCP/IP	-->TCP/-->IP
URL	Uniform Ressource Locator
WWW	World Wide Web

Literatur

[1] N.N.:
Bundesbericht Forschungsbericht 2000. http://www.bmbf.de/veroef01/ bufo2000.htm (zuletzt geprüft am 29.08.2001)

[2] N.N.:
Trend: exakte Simulation kompletter Fabriken. In: Markt- und Technik. Nr. 39/21.09.2001, Seite 38ff

[3] Wiendahl, H.-P.:
Standortbestimmung der Fabrikplanung zu Beginn des neuen Aufschwungs. Industrieschau, 41 (1995) 4, Seite 228ff

[4] Hinda, U.:
Die digitale Fabrik schafft Mehrwert. Handelsblatt, 18.04.2001

[5] N.N.:
DIN 8580: Fertigungsverfahren. Einteilung. Deutsches Institut für Normung e. V. Beuth Verlag, Berlin 1974

[6] N.N.:
VDI-Richtlinie 2525, VDI-Verlag, Düsseldorf 1999

[7] Eversheim, W.; Schuh, G. (Hrsg.):
Betriebshütte, Springer Verlag, Berlin 1999

[8] Frese, E.:
Organisationsstrukturen und Managementsystem. In Eversheim/Schuh: Betriebshütte - Produktion und Management. Springer Verlag, Berlin 1996

[9] Wedekind, H. G.; Görz, R.; Inhetveen, R.:
Zur Rolle der Informatik in einem künftigen Computational Engineering. http://www-sfb182.informatik.uni-erlangen.de/SFB182/TP/B/B4/pub (zuletzt geprüft am 29.08.2001)

[10] Schigalla, H.:
Fabrikplanung - Begriffe und Zusammenhänge. REFA-Fachbuchreihe Betriebsorganisation, Carl Hanser Verlag, München 1995

[11] Bleicher, K.:
Das Konzept integriertes Management. 3. Auflage, Campus, Frankfurt/M. 1995

[12] Collisi, T.; Fahlbusch, M.; Hagmann, M.; Ostermann, A.; Wuttke, C.C.; Weiß, M.:
Hierarchische Simulationsmodelle. Logistik für Unternehmen, 4/5 2000, VDI, Springer Verlag, Düsseldorf 2000

[13] Solvie, M.:
Zeitbehandlung und Multimedia-Unterstützung in Feldbuskommunikationssystemen. Dissertation, Universität Erlangen, Hanser Verlag, München 1995

[14] Zangemeister, C.:
Nutzwertanalyse in der Systemtechnik: Eine Methodik zur multidimensiona-

len Bewertung und Auswahl von Projektalternativen. Wittmannsche Buchhandlung, München 1976

[15] Spath, D.; Hartel, M.; Kohlmeyer,R.:
Qualitätssicherung in der Montage variantenreicher Serienprodukte. Präventiver Verfahrensansatz zur Vermeidung möglicher Montagefehler. REFA-Nachrichten, Band 49 (1996) Heft 5, Seite 13-21

[16] Law, A. M.; Kelton, W. D.:
Simulation Modeling and Analysis. 2nd Ed., McGraw Hill, New York 1991

[17] Wöhe, G.:
Einführung in die allgemeine Betriebswirtschaftslehre. 20. Auflage, Verlag Vahlen, München 2000

[18] N.N.:
VDI 3633, Blatt 1, VDI-Verlag, Düsseldorf 1993

[19] Tietz, B.:
Grundlagen der Handelnsforschung, Bd. I, Die Methoden. Rüschlikon-Zürich 1969, Seite 611

[20] Colombo, W. A.:
Development and Implementation of Hierarchical Control Structures of Flexible Production Control Systems Using High-Level Petri Nets. Dissertation Universität Erlangen, Meisenbach Verlag, Bamberg 1998

[21] Reinhart, G.; Feldmann, K.:
Simulation - Schlüsseltechnologie der Zukunft. Herbert Utz Verlag 1997

[22] Wenzel, S.; Noche, B.:
Marktspiegel - Simulation in Produktion und Logistik. TÜV Rheinland, Köln 1991

[23] Klußmann, J.; Krauth, J.; Splanemann, R.:
Simulation - Spielerei oder zukunftweisende Technik? In: wt 86(1996), Seite 361

[24] Fritsche, B.; Volger, A.:
Simulation für die Fertigungsvorbereitung. In: IM 01/1996, Seite 6

[25] Feldmann, K.; Wunderlich, J.:
Datenversorgung und Informationsbedarf des Produktionskostenmanagements - Ergebnisse einer empirischen Untersuchung. industrie management (15)99 6, Berlin 1999, Seite 48 ff

[26] Balci, O.:
How to Assess the Acceptability and Credibility of Simulation Results. In: Proceedings of the 1989 Winter Simulation Conference, IEEE 1989, Seite 62-71

[27] Fishwick, P. A.; Lee, K.:
Two Methods for Exploting Abstraction in Simulation. In: Proceedings of the AI, Simulation and Planning in high Autonomous Systems Conference. 1996, Seite 257-264

[28] Rauh, E.:
Methodische Einbindung der Simulation in die betrieblichen Planungs- und Entscheidungsabläufe. Dissertation, Universität Erlangen. Meisenbach Verlag, Bamberg 1998

[29] Schönherr, M.:
Recherche: Modellierung und Simulation als strategischer Erfolgsfaktor. In: industrie management, 16(3)2000,Berlin 2000, Seite 86-95

[30] Horn, V; Hein, J.:
Komplexe Produktionssysteme planen. ZwF, (6), 1999, Seite 300 ff

[31] Gottlob, G.; Nejdl, W.:
Expert Systems in Engineering. Principles and Applications. International Workshop. Vienna, Austria, September 24-26, 1990. Proceedings (Lecture Notes in Computer Science Vol. 462), Springer Verlag, Berlin 1990

[32] Feldmann, K.; Collisi, T., Wunderlich, J.:
Approach of a Simulation Assisted Reengineering for Manufacturing Systems. Proceedings of the 9th Simulation Symposium, October 19th-22th, 1997, Passau, Germany, Seite 421

[33] Markert, D.:
Ein neuer Ansatz zur Problematik der Datenbeschaffung, Integration und Modellbildung bei der Simulation von Materialflußsystemen. Dissertation, Universität Essen 1997

[34] Sossna, F.:
Automatische Grobsimulation zur Auslegung der Rohbau-Fertigung im Automobilbau. In: Reinhart, G.: Virtuelle Produktion. iwb Seminar, TU München, Garching 2001

[35] Duveneck, D.:
Effizienzsteigerung in der Fahrwerkfertigung mit Hilfe der Ablaufsimulation. In: Reinhart, G.: Virtuelle Produktion. iwb Seminar, TU München, Garching 2001

[36] Hartmann, B.:
Personaleinsatzplanung als Bestandteil einer integrierten ganzheitlichen Produktionsoptimierung. In: Reinhart, G.: Virtuelle Produktion. iwb Seminar, TU München, Garching 2001

[37] Schwab, J.:
Die Digitale Fabrik im Spannungsfeld zwischen Mensch und Technik. In: Reinhart, G.: Virtuelle Produktion. iwb Seminar, TU München, Garching 2001

[38] Kerzner, H.:
Project Management: A Systems Approach to Planning, Scheduling, and Controlling. 7th edition, John Wiley & Sons, New York 2000

[39] Miebach, J.:
Warum Projekte scheitern können. In: Fördern und Heben, 03/2000, Seite 181 ff

[40] Armbruster, R. Hofmann, M.:
Simulation im Lebenszyklus einer Anlage - Erprobung komplexer Strategien

in einem Kommissioniersystem. In: Simulation und Logistik. gfmt, München 1986, Seite 77

[41] Schulze, L.:
Simulation von Materialflußsystemen. Verlag Moderne Industrie, Landsberg/Lech 1988

[42] Hellbrück, G.-M.:
Der Einsatz von Simulationswerkzeugen in der praktischen Fabrikplanung. In: Simulation und Logistik. gfmt, München 1986, Seite 77

[43] Bracht, U.:
Integration der Simulation in die rechnergestützte Fabrikplanung. In: VDI Simulation von Systemen in der Logistik. Materialfluss und Produktion, VDI Verlag, Düsseldorf 1992, Seite 73

[44] Reinhart, G.; Selke, C.:
Effiziente Erstellung von Simulationsmodellen durch Integration ins informationstechnische Umfeld. In: Feldmann, K.; Reinhart, G. (Hrsg.): Simulationsbasierte Planungssysteme für Organisation und Produktion. Springer Verlag, Berlin 2000

[45] N.N.:
Modelldatenmanagement. In: Automobilindustrie, 11/2000, Seite 26 f

[46] N.N.:
Recherche: Engineering Daten Management. In: industrie management, 16(2)2000, Berlin 2000, Seite 42ff

[47] Berger, N.:
Durchgängige Daten für den Konstrukteur aufbereitet. In: Industrie Anzeiger, 19/2000, Seite 27

[48] Reinhart, G.; Brandner, S.:
Produktdaten- und Prozessmanagement in virtuellen Fabriken. In: industrie management, 16(01)2000, Berlin 2000, Seite 8 ff

[49] Hermann, T.:
Vorsprung durch Informationstransparenz. In: Werkstatt und Betrieb, 05/2000, Seite 39 ff

[50] Tanenbaum, A.:
Implementing a Corporate Repository, The Models Meet Reality. New York, John Wiley & Sons 1994

[51] Bracht, U.; Hagmann, M.:
Die ganze Fabrik im Simulationsmodell. Ein neuer Ansatz hierarchischer Gesamtmodellierung. In: ZwF, Band 93 (1998), Heft 7/8, Seite 345 ff

[52] Ortner, E.:
Von der Datenmodellierung zum Informationsmanagement. In: Müller-Ettrich, G. (Hrsg.): Fachliche Modellierung von Informationssystemen, Addison Wesley, Bonn 1993, Seite 19

[53] N.N.:
http://www.deloitte.com/whatsnew/ebusiness.html (zuletzt geprüft am 29.08.2001)

[54] N.N.:
Ovum Studie "Knowledge Management", 1998

[55] Prähofer, H.; Sametinger, J.; Stritzinger, A.:
Discrete Event Simulation using the JavaBeans Component Model. Proceedings of WEBSIM99, 1999 International Conference On Web-Based Modelling & Simulation, San Francisco, California, January 17 -20, 1999

[56] Page, E.; Buss, A.; Fishwick, P.; Healy, R.; Nance, R.; Paul, R.:
Web-Based Simulation: Evolution or Revolution? ACM Transactions on Modeling and Computer Simulation, Vol. 10, No. 1, 2000, Seite 3-17

[57] Fishwick, P. A.:
On Web-Based Models and Repositories. Enabling Technology for Simulation Science within SPIE '01 AeroSense Conference, Orlando 2001.

[58] Kuhn, T.-A.:
Struktur der wissenschaftlichen Revolution. Suhrkamp, Frankfurt 2001

[59] Klußmann, J.; Vöge, M; Krauth, J.:
Neutrales Produktdatenmodell zur Einbindung der Simulation in die betrieblichen Abläufe. wt Produktion und Management 86 (1996) 6, Berlin 1996

[60] Becker, B. D.:
SiMPLE++ goes online. In: Milberg, J.; Reinhard, G. (Hrsg.): Rationelle Nutzung der Simulationstechnik - Entwicklungstrends und Praxisbeispiele. Herbert Utz Verlag, München 1997

[61] N.N.:
VDI-Z: Marktstudie PPS/CAQ. VDI-Verlag, Düsseldorf 1996

[62] Bager, J.; Becker, J.; Munz, R.:
Zentrallager. In: c't 9/97, Heise Verlag, Hannover 1997, Seite 284 ff

[63] Hansen, O.:
Synchronisierte Simulation. Ein Beitrag zur optimalen Ablaufsteuerung von Fertigungsprozessen, dargestellt anhand der Elektronikfertigung. Dissertation, TU Dresden 1995

[64] N.N.:
VDI- Richtline 3633, Blatt 6, Gründruck, Stand Juni 1999

[65] Wunderlich, J.:
Kostensimulation - Simulationsbasierte Wirtschaftlichkeitsregelung von Produktionssystemen. Dissertation, Universität Erlangen. Meisenbach Verlag, Bamberg, (Erscheinungstermin voraussichtlich Anfang 2002)

[66] Poensgen, W.:
Vom Taylorismus zum Prozeßmanagement. In AV 06/1994

[67] Williams, J.W.:
How Simulation Gains Acceptance as a Manufacturing Productivity Improvement Tool. In Proceedings of the ESM '97, 1997, Istanbul

[68] Weule, H.:
Information als Produktionsfaktor. In: Görke, W.; Rinisland. H.; Syrbe, M.

(Hrsg.): Information als Produktionsfaktor. (GI Jahrestagung, Karlsruhe 1992). Berlin u.a: Springer Verlag, 1992, (Reihe Informatik aktuell), Seite 3 ff

[69] Abels, S.:
Modellierung und Optimierung von Montageanlagen in einem integrierten Simulationssystem. Dissertation, Universität Erlangen. Hanser Verlag, München 1993

[70] Spur, G.; Krause, F.-L.:
Das virtuelle Produkt: Management in der CAD-Technik. Hanser Verlag, München 1997

[71] Thim, C.:
Rechnergestützte Optimierung von Materialflußstrukturen in der Elektronikmontage durch Simulation. Dissertation, Universität Erlangen, Hanser Verlag, München 1992

[72] Hirschberg, A. G.; Heitmann, K.:
Simulation in German Industry - A Survey. Proceedings of the 9th Simulation Symposium, October 19th - 22nd 1997, Passau, Germany, Seite 429

[73] N.N.:
Gabler Wirtschaftslexikon. 14. Auflage. Gabler Verlag, Wiesbaden 1997, Seite 1865

[74] N.N.:
Industrial automation -- Shop floor production -- Part 2: Application of the reference model for standardization and methodology . ISO Technical Report 10314-1, ISO, Genf 1991

[75] Mertins, K.; Süssenguth, W.; Jochem, R.:
Modellierungsmethoden für rechnerintegrierte Produktionsprozesse: Unternehmensmodellierung, Softwareentwurf, Schnittstellendefinition, Simulation. Hanser Verlag, München 1994

[76] Apsel, T.:
Funktionale Dekomposition eines universell einsetzbaren Simulationssystems. In: SiP - Simulation in Passau. Universität Passau, ORS, Eigenverlag, Passau 1995, Seite 18-20

[77] N.N.:
Simulation - ein Optimierungswerkzeug für Planung und Steuerung. Logistik im Unternehmen, 6/90, Springer Verlag, Berlin 1990

[78] Biethahn, J.; Schmidt, B.:
Simulation als betriebliche Entscheidungshilfe. Springer Verlag, Berlin 1987

[79] Kormanicki, J. (Hrsg.):
Simulationstechnik. Einführung im Medienverbund. VDI-Verlag, Düsseldorf 1980, S. 167

[80] Wenzel, S.:
Modellbildung in der Simulation logistischer Systeme. industrie management, 16(3)2000, Berlin 2000, Seite 32 ff

[81] Keller, G.; Ladd, A.; Curran, T.:
Sap R/3 Business Blueprint : Understanding the Business Process Reference Model. Prentice Hall, New York 1997

[82] Wenzel, S. (Hrsg.):
Referenzmodelle für die Simulation in Produktion und Logistik. In: Fortschrittsbereichte in der Simulationstechnik. SCS Verlag, Erlangen 2000

[83] N.N.:
SEMI E10-0699 Standard for definition and measurement of equipment reliability, availability, and maintainability (RAM). Global Metrics Committee, Stand: 06/1999

[84] N.N.:
Bundesdatenschutzgesetz BDSG vom 20. Dezember 1990 (BGBl.I S. 2954, 2955), Stand: 06/1994

[85] Sachs, L.:
Angewandte Statistik. Springer Verlag, Berlin, Heidelberg, New York, Tokyo 1984

[86] Eversheim, W.:
Organisation in der Produktionstechnik. Band 3: Auftragsvorbereitung. Düsseldorf. VDI-Verlag 1980

[87] N.N.:
VDI Richtlinie 3633, Blatt 3, Entwurf, Experimentplanung und -auswertung. VDI Verlag, Düsseldorf 1997

[88] Larman, C.:
Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process. Prentice Hall, New York 2001

[89] Meyer, R.J.:
IDEF0 Functional Modelling. Knowledge Base, Texas 1992

[90] Lewis, R.:
Programming Industrial Control Systems Using IEC 1131-3. IEEE Control Engineering Series , Vol 50, Inspec, IEEE, 1998

[91] Owen, J.:
STEP - An Introduction. Information Geometers, Winchester 1993

[92] Perera, T, Liyanage, K.:
Mothodology for rapid identification and collection of input data in the simulation of manucaturing systems. In: Simulation Practice and Theory. Elevier 7 (2000), pp 645-656

[93] Feldmann, K; Collisi, T.:
Simulationsunterstützung bei der langfristigen Werkebelegungsplanung. In: Feldmann, K., Reinhart, G.: Simulationsbasierte Planungssysteme für Organisation und Planung. Springer Verlag, Berlin 2000, Seite 86 ff

[94] Mößmer, H.:
Methode zur simulationsbasierten Regelung zeitvarianter Produktionssysteme. Disseration, Forschungsbereichte iwb, Herbert Utz Verlag, München 1999

[95] Fuhrberg, K.:
Sicherheit im Internet. http://www.bsi.de/bsi-cert/index.html (zuletzt geprüft am 29.08.2001)

[96] Martens, H. :
Wirtschaft: Boom der Industriespionage. Der Spiegel, 13/99, Hamburg 1999

[97] Rescorla, E.:
SSL and TLS. Addison Wesley, Reading 2000

[98] Collisi, T.:
Hierarchische Simulationsmodelle. http://www.faps.uni-erlangen.de/persons/collisi/vdi_h_sim/index.html (zuletzt geprüft am 29.08.2001)

[99] Johnson, M. E.; Poorte, J. P.:
A hierarchical approach to computer animation in simulation modeling. In: Simulation 50:1, 1998, Seite 30-36

[100] Stoyan, H.:
Programmiermethoden der Künstlichen Intelligenz. Springer Verlag, Berlin Heidelberg 1998

[101] Rumbough, J.; Blaha, M.; Premerlani, W.; Eddy, F.; Lorensen, B.:
Object-Oriented Modeling and Design. Prentice Hall 1991

[102] Mowbray, J. T.; Malveau, R. C.:
CORBA Design Patterns. John Wiley and Sons, New York 1997

[103] Orfali, R.; Harkey, D.:
Client/Server Programming with JAVA and CORBA. John Wiley and Sons, New York 1997

[104] Firesmith, D. G.; Eykholt, E. M.:
Dictionary of Object Technology - The Definitive Desk Reference. SIGS Books, New York 1995

[105] Pidd, M.; Bayer Castro, R.:
Hierarchical Modular Modelling in Discrete Simulation. In: Proceedings of the 1998 Winter-Simulation-Conference, Washington 1998, Seite 383-389

[106] Haller, M.; Nemmer, M.:
Anforderungsgerechte Modellbildung zum projektbegleitenden Einsatz der dynamischen Materialfluss-Simulation. In: Fortschritte der Simulationstechnik, Tagungsband 11. Symposium Simulationstechnik, 1997, Seite 569-574

[107] Kistner, K.-P.; Steven, M.:
Die Bedeutung des Operations Research für die hierarchische Produktionsplanung. In: OR Spektrum (1991) 13, Seite 123-132

[108] Haddock, J.:
Automated simulation modelling and analysis for manufacturing systems. In: Production Planning & Control, Vol. 6, No. 4, 1995, Seite 352-357

[109] Booch, G.:
Object oriented design. The Benjamin/Cummings Publ. Comp., Redwood City 1991

[110] Schlögl, W.:
Integriertes Simulationsdaten-Management für Maschinenentwicklung und Anlagenplanung. Dissertation, Universität Erlangen. Meisenbach Verlag, Bamberg 2000

[111] Black Forrest Group:
Workflow Requirements of the Black Forrest Group. In: Österle, H. und Vogler, P.: Praxis des Workflow Managements, Vieweg Verlag, Wiesbaden, 1998

[112] Scheid, W.-M.; Nothnagel, B.:
Nutzen, Dauer und Kosten von PPS-Projekten in kleinen Unternehmen. In PPS Management 4 (1999) 2, Berlin 1999, Seite24 - 26

[113] Bernstein, P. A.:
Middleware. In: Communications of the ACM. Feb. 1996/ Vol. 39. No. 2, Seite 86 ff

[114] Gates, W.H.:
Digitales Business - Wettbewerb im Informationszeitalter. Microsoft Press, München 1999

[115] Charles, J.:
Middleware moves to the front, Computer 32(1999) 5, Seite 17 ff

[116] Mertens, P.; Möhle, S.; Braun, M.:
Die Entwicklung eines PPS-Systems aus Componentware. In: Wedekind, H.; Kleinöder, J.: Von der Informatik zu Computational Science und Computational Engieering. Abschlusskolloquium des SFB 182, Arbeitsberichte des IMMD, Erlangen 1998

[117] Eddon, G.; Eddon, H.:
Inside Distributed COM. Microsoft Press, Redmond 1998

[118] Ruh, W. A.; Herron, T.; Klinker, P.:
IIOP Complete: Understanding CORBA and Middleware Interoperability. Addison Wesley, Reading 1999

[119] Object Management Group:
Object Management Architecture Guide, Revision 2. John Wiley & Sons, New York 1998

[120] Hegering, H. G.; Abeck, S; Neumair, B.:
Integriertes Management vernetzter Systeme. dpunkt.verlag, Heidelberg 1999

[121] Object Management Group:
Facility for Distributed Simulation Systems. OMG document number formal/99-05-01. http://www.omg.org/cgi-bin/doc?formal/99-05-01 (Stand 1999) (zuletzt geprüft am 29.08.2001)

[122] N.N.:
IEEE draft standard P1516 (HLA Rules), P1516.1 (Interface Specification) and P1516.2 (OMT). http://standards.ieee.org (zuletzt geprüft am 29.08.2001)

[123] Sargent, R. A.; Brehon, M. J.:
Using the Combat Automation Requirement Testbed and High Level Archi-

tecture Time Management to Integrate Event-Driven and Time-Stepped Simulations - Lessons Learned. In : 2001 Advanced Simulation Technologies Conference, SCS, Seattle 2001, Seite 123 ff

[124] Tietje, H.; Straßburger, S.:
Demonstrationen von HLA-basierten verteilten Simulationen. In. Mertins, K; Rabe, M.: The New Simulation in Prodution and Logistics. 9th ASIM Dedicated Conference "Simulation in Production and Logistics", Berlin 2000. Seite 357-364

[125] Konferenzband:
Mertins, K; Rabe, M.: The New Simulation in Prodution and Logistics. 9th ASIM Dedicated Conference "Simulation in Production and Logistics", Berlin 2000

[126] Konferenzband:
2001 Advanced Simulation Technologies Conference, SCS, Seattle 2001

[127] Konferenzband:
Simulation und Visualisierung 2001. Otto von Guericke Universität, Madgeburg 2001

[128] N.N.:
Dotnet macht Windows-Programme zu Legacy Code. In: Computer Zeitung Nr. 32/9. August 2001

[129] N.N.:
http://java.sun.com/products/jdk/1.2/index.html (zuletzt geprüft am 29.08.2001)

[130] Orfali, R.; Harkey, D.:
Client/Server Programming with Java and CORBA. John Wiley & Sons, New York 1997, Seite 185-379

[131] Norman, R.J.:
CORBA and DCOM - Side by Side. In: http://www-rohan.sdsu.edu/faculty/rnorman/papers/corba-dcom.pdf (zuletzt geprüft am 29.08.2001)

[132] B. Page et al:
Objektorientierte Simulation in JAVA mit dem Framework Desmo-J. In: Libri Books on Demand, 2000

[133] Griffel, F.:
Einsatzmöglichkeiten komponentenbasierter Meta-Modelle zur generativen, konsistenzsichernden Softwarekonstruktion. In: Tagungsband Modellierung 2000, Fölbach Verlag 2000

[134] N.N.:
http://www.omg.org/technology/documents/formal/corbamanufacturingspecs.htm (zuletzt geprüft am 29.08.2001)

[135] Bachmann, R.:
HLA und CORBA: Partner oder Konkurrenten? In Möller, D.P.F. (Hrsg): SCS, Frontiers in Simulation, 14. Symposium. Hamburg Forschrittsberichte Simulation. SCS Verlag, Ghent 2000

[136] Buss, A.; Jackson, L.:
Distributed Simulation Modelling: A Comparison of HLA, CORBA, AND RMI. In: Proceedings of the 1998 Winter Simulation Conference, 1998, Seite 819-825

[137] Stöckel, T.:
Kommunikationstechnische Integration der Prozessebene in Produktionssysteme durch Middleware-Frameworks. Dissertation, Universität Erlangen, Meisenbach Verlag, Bamberg 2000

[138] Feldmann, K.; Stöckel, T.:
A Jini-based for accessing process level data in manufacturing systems. In WGP Annals, VII/1, WGP Eigenverlag, Berlin 2000

[139] Feldmann, K.; Stöckel, T.:
Information Systems Architecture for Collaborative Manufacturing in Virtual Enterprises. In: Jacucci, G. (Hrsg.): Proceedings of the Prolamat 1998: Globalisation of Manufacturing in the Digital Communications-era of the 21st Century, Trento 1998

[140] Feldmann, K.; Stöckel, T.; Haberstumpf, B.:
Conception and Implementation of an Object Request Broker for the Integration of the Process Level in Manufacturing Systems . In: Journal of Systems Integration (2001) 10, Kluwer Academic Publishers, Boston 2001

[141] Wenzel, S; Noche, B.:
Simulationsinstrumente in Produktion und Logistik - eine Marktübersicht. In. Mertins, K; Rabe, M.: The New Simulation in Production and Logistics. 9th A-SIM Dedicated Conference "Simulation in Production and Logistics", Berlin 2000, Seite 423 ff

[142] Isermann, R.:
Modellgestützte Überwachung und Fehlerdiagnose Technischer Systeme, Teile 1 und 2. In: Automatisierungstechnische Praxis 38, Oldenbourg Verlag, Heft 5/6 1996, Seite 2 ff (48ff)

[143] Bolch, G.; Seidel, M.-M.; Vollrath, M.-M.:
Prozessautomatisierung. Teubner Verlag, Stuttgart 1993

[144] Kreyzig, E.:
Statistische Methoden und ihre Anwendungen. 7. Auflage, Vandenh. u. R., Göttingen 1999, Seite 125

[145] Foata, D.; Fuchs. A.:
Wahrscheinlichkeitsrechnung. Birkhäuser Verlag, Basel 1999

[146] Pfanznagl, J.:
Elementare Wahrscheinlichkeitsrechnung. de Gruyter, Berlin 1991

[147] Kreyzig, E.:
Advanced Engineering Mathematics. 8th Ed., John Wiley & Sons, New York 1999

[148] Rothaupt, A.:
Modulares Planungssystem zur Optimierung der Elektronikfertigung. Dissertation, Universität Erlangen, Hanser Verlag, München 1995

[149] Röhm, E.:
Auswahl von Leitständen zur Produktionssteuerung auf der Basis dynamischer Systemvergleiche. Dissertation, TH Aachen, Shaker Verlag, Aachen 1997

[150] Göhringer, J.:
Integrierte Telediagnose via Internet zum effizienten Service von Produktionssystemen. Dissertation, Universität Erlangen, Meisenbach Verlag, Bamberg 2001

[151] Schmerler, S.; Tanurhan, Y.; Müller-Glaser, K.-D.:
Predictive Time Warp. In: Kaylan, A.R., Lehmann, A. Proceedings of the 11th European Simulation Multiconference, ESM 97, 1-4 June 1997, Istanbul 1997, Seite 452 ff

[152] Chandy, K.; Misra, J.:
Distributed Simulation: Asynchronous Distributed Simulation via a Sequence of Parallel Computations. Communications of the ACM, 24(11), 1981, Seite 198ff

Lebenslauf

Thomas Collisi

geboren am 21. November 1967 in Leverkusen

verheiratet

09/74 - 08/78	Grundschule in Leverkusen
09/78 - 06/88	Landrat-Lucas-Gymnasium in Leverkusen
10/88 - 07/96	Studium der Informatik an der Universität Erlangen-Nürnberg Abschluss: Dipl.-Inf. Univ.
09/96 - 01/02	Wissenschaftlicher Mitarbeiter am Lehrstuhl für Fertigungsautomatisierung und Produktionssystematik der Universität Erlangen-Nürnberg Leiter: Prof. Dr.-Ing. K. Feldmann
10/00 - 01/02	Geschäftsführer des Bayerischen Forschungsverbundes Simulationstechnik (FORSIM)

Sonderveröffentlichungen Meisenbach Verlag, Bamberg:

Frank Vollertsen
Laserstrahlumformen, lasergestützte Formgebung: Verfahren, Mechanismen, Modellierung
252 Seiten, 110 Bilder, Tabellen und Sachregister.
1996. Hardcover
Euro 50,--
ISBN 3-87525-071-0

Ulf Engel
Beanspruchung und Beanspruchbarkeit von Werkzeugen der Massivumformung
193 Seiten, 72 Bilder, Tabellen und Sachregister.
1996. Hardcover
Euro 50,--
ISBN 3-87525-072-9

Reihe Fertigungstechnik - Erlangen

Band 1 - 52
Carl Hanser Verlag, München

ab Band 53
Meisenbach Verlag, Bamberg
45,-- Euro

Band 1: Andreas Hemberger
Innovationspotentiale in der rechnerintegrierten Produktion durch wissensbasierte Systeme
208 Seiten, 107 Bilder. 1988.

Band 2: Detlef Classe
Beitrag zur Steigerung der Flexibilität automatisierter Montagesysteme durch Sensorintegration und erweiterte Steuerungskonzepte
194 Seiten, 70 Bilder. 1988.

Band 3: Friedrich–Wilhelm Nolting
Projektierung von Montagesystemen
201 Seiten, 107 Bilder, 1 Tabelle. 1989.

Band 4: Karsten Schlüter
Nutzungsgradsteigerung von Montagesystemen durch den Einsatz der Simulationstechnik
177 Seiten, 97 Bilder. 1989.

Band 5: Shir–Kuan Lin
Aufbau von Modellen zur Lageregelung von Industrierobotern
168 Seiten, 46 Bilder. 1989.

Band 6: Rudolf Nuss
Untersuchungen zur Bearbeitungsqualität im Fertigungssystem Laserstrahlschneiden
206 Seiten, 115 Bilder, 6 Tabellen. 1989.

Band 7: Wolfgang Scholz
Modell zur datenbankgestützten Planung automatisierter Montageanlagen
194 Seiten, 89 Bilder. 1989.

Band 8: Hans–Jürgen Wißmeier
Beitrag zur Beurteilung des Bruchverhaltens von Hartmetall–Fließpreßmatrizen
179 Seiten, 99 Bilder, 9 Tabellen. 1989.

Band 9: Rainer Eisele
Konzeption und Wirtschaftlichkeit von Planungssystemen in der Produktion
183 Seiten, 86 Bilder. 1990.

Band 10: Rolf Pfeiffer
Technologisch orientierte Montageplanung am Beispiel der Schraubtechnik
216 Seiten, 102 Bilder, 16 Tabellen. 1990.

Band 11: Herbert Fischer
Verteilte Planungssysteme zur Flexibilitätssteigerung der rechnerintegrierten Teilefertigung
201 Seiten, 82 Bilder. 1990.

Band 12: Gerhard Kleineidam
CAD/CAP: Rechnergestützte Montagefeinplanung
203 Seiten, 107 Bilder. 1990.

Band 13: Frank Vollertsen
Pulvermetallurgische Verarbeitung
eines übereutektoiden verschleißfesten Stahls
XIII u. 217 Seiten, 67 Bilder, 34 Tabellen. 1990.

Band 14: Stephan Biermann
Untersuchungen zur Anlagen– und Prozeßdiagnostik
für das Schneiden mit CO_2–Hochleistungslasern
VIII u. 170 Seiten, 93 Bilder, 4 Tabellen. 1991.

Band 15: Uwe Geißler
Material– und Datenfluß
in einer flexiblen Blechbearbeitungszelle
124 Seiten, 41 Bilder, 7 Tabellen. 1991.

Band 16: Frank Oswald Hake
Entwicklung eines rechnergestützten Diagnosesystems
für automatisierte Montagezellen
XIV u. 166 Seiten, 77 Bilder. 1991.

Band 17: Herbert Reichel
Optimierung der Werkzeugbereitstellung
durch rechnergestützte Arbeitsfolgenbestimmung
198 Seiten, 73 Bilder, 2 Tabellen. 1991.

Band 18: Josef Scheller
Modellierung und Einsatz von Softwaresystemen
für rechnergeführte Montagezellen
198 Seiten, 65 Bilder. 1991.

Band 19: Arnold vom Ende
Untersuchungen zum Biegeumformen mit elastischer Matrize
166 Seiten, 55 Bilder, 13 Tabellen. 1991.

Band 20: Joachim Schmid
Beitrag zum automatisierten Bearbeiten
von Keramikguß mit Industrierobotern
XIV u. 176 Seiten, 111 Bilder, 6 Tabellen. 1991.

Band 21: Egon Sommer
Multiprozessorsteuerung für kooperierende
Industrieroboter in Montagezellen
188 Seiten, 102 Bilder. 1991.

Band 22: Georg Geyer
Entwicklung problemspezifischer Verfahrensketten
in der Montage
192 Seiten, 112 Bilder. 1991.

Band 23: Rainer Flohr
Beitrag zur optimalen Verbindungstechnik
in der Oberflächenmontage (SMT)
186 Seiten, 79 Bilder. 1991.

Band 24: Alfons Rief
Untersuchungen zur Verfahrensfolge Laserstrahlschneiden
und –schweißen in der Rohkarosseriefertigung
VI u. 145 Seiten, 58 Bilder, 5 Tabellen. 1991.

Band 25: Christoph Thim
Rechnerunterstützte Optimierung von Materialflußstrukturen
in der Elektronikmontage durch Simulation
188 Seiten, 74 Bilder. 1992.

Band 26: Roland Müller
CO_2-Laserstrahlschneiden
von kurzglasverstärkten Verbundwerkstoffen
141 Seiten, 107 Bilder, 4 Tabellen. 1992.
Band 27: Günther Schäfer
Integrierte Informationsverarbeitung bei der Montageplanung
195 Seiten, 76 Bilder. 1992.

Band 28: Martin Hoffmann
Entwicklung einer CAD/CAM–Prozeßkette
für die Herstellung von Blechbiegeteilen
149 Seiten, 89 Bilder. 1992.

Band 29: Peter Hoffmann
Verfahrensfolge Laserstrahlschneiden und –schweißen : Prozeßführung und Systemtechnik in der 3D–Laserstrahlbearbeitung von Blechformteilen
186 Seiten, 92 Bilder, 10 Tabellen. 1992.

Band 30: Olaf Schrödel
Flexible Werkstattsteuerung mit objektorientierten Softwarestrukturen
180 Seiten, 84 Bilder. 1992.

Band 31: Hubert Reinisch
Planungs– und Steuerungswerkzeuge zur impliziten Geräteprogrammierung in Roboterzellen
XI u. 212 Seiten, 112 Bilder. 1992.

Band 32: Brigitte Bärnreuther
Ein Beitrag zur Bewertung des Kommunikationsverhaltens von Automatisierungsgeräten in flexiblen Produktionszellen
XI u. 179 Seiten, 71 Bilder. 1992.

Band 33: Joachim Hutfless
Laserstrahlregelung und Optikdiagnostik in der Strahlführung einer CO_2-Hochleistungslaseranlage
175 Seiten, 70 Bilder, 17 Tabellen. 1993.

Band 34: Uwe Günzel
Entwicklung und Einsatz eines Simulationsverfahrens für operative und strategische Probleme der Produktionsplanung und –steuerung
XIV u. 170 Seiten, 66 Bilder, 5 Tabellen. 1993.

Band 35: Bertram Ehmann
Operatives Fertigungscontrolling durch Optimierung auftragsbezogener Bearbeitungsabläufe in der Elektronikfertigung
XV u. 167 Seiten, 114 Bilder. 1993.

Band 36: Harald Kolléra
Entwicklung eines benutzerorientierten Werkstattprogrammiersystems für das Laserstrahlschneiden
129 Seiten, 66 Bilder, 1 Tabelle. 1993.

Band 37: Stephanie Abels
Modellierung und Optimierung von Montageanlagen in einem integrierten Simulationssystem
188 Seiten, 88 Bilder. 1993.

Band 38: Robert Schmidt–Hebbel
Laserstrahlbohren durchflußbestimmender Durchgangslöcher
145 Seiten, 63 Bilder, 11 Tabellen. 1993.

Band 39: Norbert Lutz
Oberflächenfeinbearbeitung keramischer Werkstoffe mit XeCl–Excimerlaserstrahlung
187 Seiten, 98 Bilder, 29 Tabellen. 1994.

Band 40: Konrad Grampp
Rechnerunterstützung bei Test und Schulung an Steuerungssoftware von SMD–Bestücklinien
178 Seiten, 88 Bilder. 1995.

Band 41: Martin Koch
Wissensbasierte Unterstützung der Angebotsbearbeitung in der Investitionsgüterindustrie
169 Seiten, 68 Bilder. 1995.

Band 42: Armin Gropp
Anlagen– und Prozeßdiagnostik beim Schneiden mit einem gepulsten Nd:YAG–Laser
160 Seiten, 88 Bilder, 7 Tabellen. 1995.

Band 43: Werner Heckel
Optische 3D–Konturerfassung und on–line Biegewinkelmessung mit dem Lichtschnittverfahren
149 Seiten, 43 Bilder, 11 Tabellen. 1995.

Band 44: Armin Rothhaupt
Modulares Planungssystem zur Optimierung der Elektronikfertigung
180 Seiten, 101 Bilder. 1995.

Band 45: Bernd Zöllner
Adaptive Diagnose in der Elektronikproduktion
195 Seiten, 74 Bilder, 3 Tabellen. 1995.

Band 46: Bodo Vormann
Beitrag zur automatisierten Handhabungsplanung komplexer Blechbiegeteile
126 Seiten, 89 Bilder, 3 Tabellen. 1995.

Band 47: Peter Schnepf
Zielkostenorientierte Montageplanung
144 Seiten, 75 Bilder. 1995.

Band 48: Rainer Klotzbücher
Konzept zur rechnerintegrierten Materialversorgung in flexiblen Fertigungssystemen
156 Seiten, 62 Bilder. 1995.

Band 49: Wolfgang Greska
Wissensbasierte Analyse und Klassifizierung von Blechteilen
144 Seiten, 96 Bilder. 1995.

Band 50: Jörg Franke
Integrierte Entwicklung neuer Produkt- und Produktionstechnologien für räumliche spritzgegossene Schaltungsträger (3-D MID)
196 Seiten, 86 Bilder, 4 Tabellen. 1995.

Band 51: Franz-Josef Zeller
Sensorplanung und schnelle Sensorregelung für Industrieroboter
190 Seiten, 102 Bilder, 9 Tabellen. 1995.

Band 52: Michael Solvie
Zeitbehandlung und Multimedia-Unterstützung in Feldkommunikationssystemen
200 Seiten, 87 Bilder, 35 Tabellen. 1996.

Band 53: Robert Hopperdietzel
Reengeneering in der Elektro- und Elektronikindustrie
180 Seiten, 109 Bilder, 1 Tabelle. 1996.
ISBN 3-87525-070-2

Band 54: Thomas Rebhan
Beitrag zur Mikromaterialbearbeitung mit Excimerlasern – Systemkomponenten und Verfahrensoptimierungen
148 Seiten, 61 Bilder, 10 Tabellen. 1996.
ISBN 3-87525-075-3

Band 55: Henning Hanebuth
Laserstrahlhartlöten mit Zweistrahltechnik
157 Seiten, 58 Bilder, 11 Tabellen. 1996.
ISBN 3-87525-074-5

Band 56: Uwe Schönherr
Steuerung und Sensordatenintegration für flexible Fertigungszellen mit kooperierenden Robotern
188 Seiten, 116 Bilder, 3 Tabellen. 1996.
ISBN 3-87525-076-1

Band 57: Stefan Holzer
Berührungslose Formgebung mit Laserstrahlung
162 Seiten, 69 Bilder, 11 Tabellen. 1996.
ISBN 3-87525-079-6

Band 58: Markus Schultz
Fertigungsqualität beim 3D-Laserstrahlschweißen von Blechformteilen
165 Seiten, 88 Bilder, 9 Tabellen. 1997.
ISBN 3-87525-080-X

Band 59: Thomas Krebs
Integration elektromechanischer CA-Anwendungen über einem STEP-Produktmodell
198 Seiten, 58 Bilder, 8 Tabellen. 1997.
ISBN 3-87525-081-8

Band 60: Jürgen Sturm
Prozeßintegrierte Qualitätssicherung
in der Elektronikproduktion
167 Seiten, 112 Bilder, 5 Tabellen. 1997.
ISBN 3-87525-082-6

Band 61: Andreas Brand
Prozesse und Systeme zur Bestückung
räumlicher elektronischer Baugruppen (3D-MID)
182 Seiten, 100 Bilder. 1997.
ISBN 3-87525-087-7

Band 62: Michael Kauf
Regelung der Laserstrahlleistung und der Fokusparameter
einer CO_2-Hochleistungslaseranlage
140 Seiten, 70 Bilder, 5 Tabellen. 1997.
ISBN 3-87525-083-4

Band 63: Peter Steinwasser
Modulares Informationsmanagement
in der integrierten Produkt– und Prozeßplanung
190 Seiten, 87 Bilder. 1997.
ISBN 3-87525-084-2

Band 64: Georg Liedl
Integriertes Automatisierungskonzept
für den flexiblen Materialfluß in der Elektronikproduktion
196 Seiten, 96 Bilder, 3 Tabellen. 1997.
ISBN 3-87525-086-9

Band 65: Andreas Otto
Transiente Prozesse beim Laserstrahlschweißen
132 Seiten, 62 Bilder, 1 Tabelle. 1997.
ISBN 3-87525-089-3

Band 66: Wolfgang Blöchl
Erweiterte Informationsbereitstellung an offenen CNC–Steuerungen
zur Prozeß– und Programmoptimierung
168 Seiten, 96 Bilder. 1997.
ISBN 3-87525-091-5

Band 67: Klaus–Uwe Wolf
Verbesserte Prozeßführung und Prozeßplanung
zur Leistungs– und Qualitätssteigerung beim Spulenwickeln
186 Seiten, 125 Bilder. 1997.
ISBN 3-87525-092-3

Band 68: Frank Backes
Technologieorientierte Bahnplanung für die 3D–Laserstrahlbearbeitung
138 Seiten, 71 Bilder, 2 Tabellen. 1997.
ISBN 3-87525-093-1

Band 69: Jürgen Kraus
Laserstrahlumformen von Profilen
137 Seiten, 72 Bilder, 8 Tabellen. 1997.
ISBN 3-87525-094-X

Band 70: Norbert Neubauer
Adaptive Strahlführungen für CO_2-Laseranlagen
120 Seiten, 50 Bilder, 3 Tabellen. 1997.
ISBN 3-87525-095-8

Band 71: Michael Steber
Prozeßoptimierter Betrieb flexibler Schraubstationen
in der automatisierten Montage
168 Seiten, 78 Bilder, 3 Tabellen. 1997.
ISBN 3-87525-096-6

Band 72: Markus Pfestorf
Funktionale 3D–Oberflächenkenngrößen in der Umformtechnik
162 Seiten, 84 Bilder, 15 Tabellen. 1997.
ISBN 3-87525-097-4

Band 73: Volker Franke
Integrierte Planung und Konstruktion von Werkzeugenfür die Biegebearbeitung
143 Seiten, 81 Bilder. 1998.
ISBN 3-87525-098-2

Band 74: Herbert Scheller
Automatisierte Demontagesysteme und recyclinggerechte Produktgestaltung elektronischer Baugruppen
184 Seiten, 104 Bilder, 17 Tabellen. 1998.
ISBN 3-87525-099-0

Band 75: Arthur Meßner
Kaltmassivumformung metallischer Kleinstteile – Werkstoffverhalten, Wirkflächenreibung, Prozeßauslegung
164 Seiten, 92 Bilder, 14 Tabellen. 1998.
ISBN 3-87525-100-8

Band 76: Mathias Glasmacher
Prozeß– und Systemtechnik zum Laserstrahl–Mikroschweißen
184 Seiten, 104 Bilder, 12 Tabellen. 1998.
ISBN 3-87525-101-6

Band 77: Michael Schwind
Zerstörungsfreie Ermittlung mechanischer Eigenschaften von Feinblechen mit dem Wirbelstromverfahren
124 Seiten, 68 Bilder, 8 Tabellen. 1998.
ISBN 3-87525-102-4

Band 78: Manfred Gerhard
Qualitätssteigerung in der Elektronikproduktion durch Optimierung der Prozeßführung beim Löten komplexer Baugruppen
179 Seiten, 113 Bilder, 7 Tabellen. 1998.
ISBN 3-87525-103-2

Band 79: Elke Rauh
Methodische Einbindung der Simulation in die betrieblichen Planungs– und Entscheidungsabläufe
192 Seiten, 114 Bilder, 4 Tabellen. 1998.
ISBN 3-87525-104-0

Band 80: Sorin Niederkorn
Meßeinrichtung zur Untersuchung der Wirkflächenreibung bei umformtechnischen Prozessen
99 Seiten, 46 Bilder, 6 Tabellen. 1998.
ISBN 3-87525-105-9

Band 81: Stefan Schuberth
Regelung der Fokuslage beim Schweißen mit CO_2–Hochleistungslasern unter Einsatz von adaptiven Optiken
140 Seiten, 64 Bilder, 3 Tabellen. 1998.
ISBN 3-87525-106-7

Band 82: Armando Walter Colombo
Development and Implementation of Hierarchical Control Structures of Flexible Production Systems Using High Level Petri Nets
216 Seiten, 86 Bilder. 1998.
ISBN 3-87525-109-1

Band 83: Otto Meedt
Effizienzsteigerung bei Demontage und Recycling durch flexible Demontagetechnologien und optimierte Produktgestaltung
186 Seiten, 103 Bilder. 1998.
ISBN 3-87525-108-3

Band 84: Knuth Götz
Modelle und effiziente Modellbildung zur Qualitätssicherung in der Elektronikproduktion
212 Seiten, 129 Bilder, 24 Tabellen. 1998.
ISBN 3-87525-112-1

Band 85: Ralf Luchs
Einsatzmöglichkeiten leitender Klebstoffe zur zuverlässigen Kontaktierung elektronischer Bauelemente in der SMT
176 Seiten, 126 Bilder, 30 Tabellen. 1998.
ISBN 3-87525-113-7

Band 86: Frank Pöhlau
Entscheidungsgrundlagen zur Einführung räumlicher spritzgegossener Schaltungsträger (3–D MID)
144 Seiten, 99 Bilder. 1999.
ISBN 3-87525-114-8

Band 87: Roland T. A. Kals
Fundamentals on the miniaturization of sheet metal working processes
128 Seiten, 58 Bilder, 11 Tabellen. 1999.
ISBN 3-87525-115-6

Band 88: Gerhard Luhn
Implizites Wissen und technisches Handeln am Beispiel der Elektronikproduktion.
252 Seiten, 61 Bilder, 1 Tabelle. 1999.
ISBN 3-87525-116-4

Band 89: Axel Sprenger
Adaptives Streckbiegen von Aluminium–Strangpreßprofilen
114 Seiten, 63 Bilder, 4 Tabellen. 1999.
ISBN 3-87525-117-2

Band 90: Hans–Jörg Pucher
Untersuchungen zur Prozeßfolge Umformen, Bestücken und Laserstrahllöten von Mikrokontakten
158 Seiten, 69 Bilder, 9 Tabellen. 1999.
ISBN 3-87525-119-9

Band 91: Horst Arnet
Profilbiegen mit kinematischer Gestalterzeugung
128 Seiten, 67 Bilder, 7 Tabellen. 1999.
ISBN 3-87525-120-2

Band 92: Doris Schubart
Prozeßmodellierung und Technologieentwicklung beim Abtragen mit CO_2-Laserstrahlung
133 Seiten, 57 Bilder, 13 Tabellen. 1999.
ISBN 3-87525-122-9

Band 93: Adrianus L. P. Coremans
Laserstrahlsintern von Metallpulver – Prozeßmodellierung, Systemtechnik, Eigenschaften laserstrahlgesinterter Metallkörper
184 Seiten, 108 Bilder, 12 Tabellen. 1999.
ISBN 3-87525-124-5

Band 94: Hans-Martin Biehler
Optimierungskonzepte für Qualitätsdatenverarbeitung und Informationsbereitstellung in der Elektronikfertigung
194 Seiten, 105 Bilder. 1999.
ISBN 3-87525-126-1

Band 95: Wolfgang Becker
Oberflächenausbildung und tribologische Eigenschaften excimerlaserstrahlbearbeiteter Hochleistungskeramiken
175 Seiten, 71 Bilder, 3 Tabellen. 1999.
ISBN 3-87525-127-x

Band 96: Philipp Hein
Innenhochdruck-Umformen von Blechpaaren: Modellierung, Prozeßauslegung und Prozeßführung
129 Seiten, 57 Bilder, 7 Tabellen. 1999.
ISBN 3-87525-128-8

Band 97: Gunter Beitinger
Herstellungs- und Prüfverfahren für thermoplastische Schaltungsträger
169 Seiten, 92 Bilder, 20 Tabellen. 1999.
ISBN 3-87525-129-6

Band 98: Jürgen Knoblach
Beitrag zur rechnerunterstützten verursachungsgerechten Angebotskalkulation von Blechteilen mit Hilfe wissensbasierter Methoden
155 Seiten, 53 Bilder, 26 Tabellen. 1999.
ISBN 3-87525-130-X

Band 99: Frank Breitenbach
Bildverarbeitungssystem zur Erfassung der Anschlußgeometrie elektronischer SMT-Bauelemente
147 Seiten, 92 Bilder, 12 Tabellen. 2000.
ISBN 3-87525-131-8

Band 100: Bernd Falk
Simulationsbasierte Lebensdauervorhersage für Werkzeuge der Kaltmassivumformung
134 Seiten, 44 Bilder, 15 Tabellen. 2000.
ISBN 3-87525-136-9

Band 101: Wolfgang Schlögl
Integriertes Simulationsdaten-Management für Maschinenentwicklung und Anlagenplanung
169 Seiten, 101 Bilder, 20 Tabellen. 2000.
ISBN 3-87525-137-7

Band 102: Christian Hinsel
Ermüdungsbruchversagen hartstoffbeschichteter Werkzeugstähle in der Kaltmassivumformung
130 Seiten, 80 Bilder, 14 Tabellen. 2000.
ISBN 3-87525-138-5

Band 103: Stefan Bobbert
Simulationsgestützte Prozessauslegung für das Innenhochdruck-Umformen von Blechpaaren
123 Seiten, 77 Bilder. 2000.
ISBN 3-87525-145-8

Band 104: Harald Rottbauer
Modulares Planungswerkzeug zum Produktionsmanagementin der Elektronikproduktion
166 Seiten, 106 Bilder. 2001.
ISBN 3-87525-139-3

Band 105: Thomas Hennige
Flexible Formgebung von Blechen durch Laserstrahlumformen
119 Seiten, 50 Bilder. 2001.
ISBN 3-87525-140-7

Band 106: Thomas Menzel
Wissensbasierte Methoden für die rechnergestützte Charakterisierung und Bewertung innovativer Fertigungsprozesse
152 Seiten, 71 Bilder. 2001.
ISBN 3-87525-142-3

Band 107: Thomas Stöckel
Kommunikationstechnische Integration der Prozeßebene in Produktionssysteme durch Middleware-Frameworks
147 Seiten, 65 Bilder, 5 Tabellen. 2001.
ISBN 3-87525-143-1

Band 108: Frank Pitter
Verfügbarkeitssteigerung von Werkzeugmaschinen durch Einsatz mechatronischer Sensorlösungen
158 Seiten, 131 Bilder, 8 Tabellen. 2001.
ISBN 3-87525-144-X

Band 109: Markus Korneli
Integration lokaler CAP-Systeme in einen globalen Fertigungsdatenverbund
121 Seiten, 53 Bilder, 11 Tabellen. 2001.
ISBN 3-87525-146-6

Band 110
Burkhard Müller
Laserstrahljustieren mit Excimer-Lasern – Prozeßparameter und Modelle zur Aktorkonstruktion
128 Seiten, 36 Bilder, 9 Tabellen. 2001
ISBN 3-87525-159-8

Band 111: Jürgen Göhringer
Integrierte Telediagnose via Internet
zum effizienten Service von Produktionssystemen
178 Seiten, 98 Bilder, 5 Tabellen. 2001.
ISBN 3-87525-147-4

Band 112: Robert Feuerstein
Qualitäts- und kosteneffiziente Integration
neuer Bauelementetechnologien
in die Flachbaugruppenfertigung
161 Seiten, 99 Bilder, 10 Tabellen. 2001.
ISBN 3-87525-151-2

Band 113: Marcus Reichenberger
Eigenschaften und Einsatzmöglichkeiten
alternativer Elektroniklote
in der Oberflächenmontage (SMT)
165 Seiten, 97 Bilder, 18 Tabellen. 2001.
ISBN 3-87525-152-0

Band 114
Alexander Huber
Justieren vormontierter Systeme mit dem Nd:YAG-Laser
unter Einsatz von Aktoren
122 Seiten, 58 Bilder, 5 Tabellen. 2001.
ISBN 3-87525-153-9

Band 115
Sami Krimi
Analyse und Optimierung von Montagesystemen
in der Elektronikproduktion
155 Seiten, 88 Bilder, 3 Tabellen. 2001.
ISBN 3-87525-157-1

Band 116
Marion Merklein
Laserstrahlumformen von Aluminiumwerkstoffen -
Beeinflussung der Mikrostruktur und der mechanischen Eigenschaften
122 Seiten, 65 Bilder, 15 Tabellen. 2001.
ISBN 3-87525-156-3

Band 117
Thomas Collisi
Ein informationslogistisches Architekturkonzept
zur Akquisition simulationsrelevanter Daten
181 Seiten, 105 Bilder, 7 Tabellen. 2002.
ISBN 3-87525-164-4