Lehrstuhl für
Werkzeugmaschinen und Fertigungstechnik
der Technischen Universität München

Konzeption und System einer Integrationsplattform zur Entwicklung von Werkzeugmaschinen

Bernd Lercher

Vollständiger Abdruck der von der Fakultät für Maschinenwesen der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender : Univ.-Prof. Dr. rer. nat. H. Bubb

Prüfer der Dissertation:

1. Univ.-Prof. Dr.-Ing. M. Zäh
2. Univ.-Prof. Dr. rer. nat. J. Schlichter

Die Dissertation wurde am 07.01.2008 bei der Technischen Universität München eingereicht und durch die Fakultät für Maschinenwesen am 15.04.2008 angenommen.

Inhaltsverzeichnis

Verzeichnis der Formelzeichen

Große lateinische Buchstaben

Symbol	Bedeutung
$\forall$	Allquantor der Prädikatenlogik
Ap_i	Applikationsmuster
D_q	Datenquelle
E	Menge der Kanten eines Graphen
$\exists$	Existenzquantor der Prädikatenlogik
G	Graph
I	Interpretationsfunktion
$\bar{I}$	Modell einer Ontologie
M	Mediator
M^{vmt}	Mediator des Metadaten-Management-Systems
O	Ontologie
O^c	Ontologie des Kerns
O^{vmt}	Ontologie des Mediators M^{vmt}
P	Attributmenge
T	Terminologie
T^{vmt}	Terminologie des Mediators M^{vmt}
$\bigcup U$	Vereinigungsmenge aller Mengen aus U
$A \cup B$	Vereinigungsmenge
V	Menge der Knoten eines Graphen
$A \times B$	Kartesisches Produkt

Formelverzeichnis

Kleine lateinische Buchstaben

a_i	Artikulation
data	Menge der Daten
r	Wurzelknoten eines Graphen
ref	Referenzfunktion
q	Datenanfrage an eine Datenquelle
t	Ein Element der Menge *T*
v	Logischer Operator *oder*
$\wedge$	Logischer Operator *und*

Kleine griechische Buchstaben

ε	Symbolisiert eine leere Datenanfrage
ν	Attributfunktion

Symbole

$\neg$	Negation einer Aussage
$\setminus$	Differenz zweier Mengen
$\preceq$	Zuordnungsrelation
$\sim$	Äquivalenzrelation
$\in$	Element einer Menge
$\notin$	Nicht-Element einer Menge

Abkürzungsverzeichnis

Symbol	Bedeutung
AAM	Application Activity Model
ABS	Antiblockiersystem
AIM	Application Interpreted Model
AP	Application Protocol
API	Application Programming Interface
ARM	Application Reference Model
ASR	Antischlupfregelung
ATS	Application Transaction Set
AWL	Anweisungsliste
BMBF	Bundesministerium für Bildung und Forschung
BKM	Bayerisches Kompetenznetzwerk Mechatronik
BMK	Betriebsmittelkennzahl
CAD	Computer Aided Design
CACSD	Computer Aided Control System Design
CMMI	Capability Maturity Model Integration
CORBA	Common Object Request Broker Architecture
CRM	Customer Relationship Management
DESINA	Dezentralisierte standardisierte Installationstechnik
DIN	Deutsches Institut für Normung e.V.
DML	Descriptive Markup Language
DoD	US Department of Defense

DOM	Document Object Model
DTD	Document Type Definition
DV	Datenverarbeitung
EA	Eingang-Ausgang
E/A	Eingang/Ausgang
ECAD	Elektro-CAD
EBS	Elektronisches Bremssystem
EDM	Engineering Data Management
ERP	Enterprise Ressource Planing
FEM	Finite-Elemente-Methode
FIA	Föderale Informations-Architektur
HMI	Human Machine Interface
HNC	Hydraulic numerical control
IDL	Interface Description Language
IEC	International Electrotechnical Commission
IL	Instruction List
ISO	International Organization for Standardization
iViP	Integrierte virtuelle Produktentstehung
iwb	Institut für Werkzeugmaschinen und Betriebswissenschaften
KMU	Kleine und mittlere Unternehmen
MDR	Metadata Repository
MKS	Mehrkörpersimulation
MTM	Machine Tool Metamodell
MOF	Meta Object Facility

NC	Numerical Control
OASIS	Org. for the Advancement of Structured Information Standards
OCL	Object Constraint Language
ODF	Open Document Format for Office Applications
OEM	Object Exchange Model
OMG	Object Management Group
PDI	Product Data Interoperability
PDM	Produktdatenmanagement
PDML	Product Data Markup Language
PID	Proportional-Integral-Differential
PLC	Programmable Logic Control
PLM	Product Life Cycle Management
PML	Procedural Markup Language
POE	Programmorganisationseinheiten
RIF	Requirements Interchange Format
ROOM	Realtime Object Oriented Modeling
SCM	Software Configuration Management, Supply Chain Management
SDAI	Standard Data Access Interface
SFC	Sequential Function Charts (dt. Schrittkette)
SPICE	Software Process Improvement and Capability Determination
SPS	Speicherprogrammierbare Steuerung
ST	Structured Text
STEP	Standard for the Exchange of Product Data
TDM	Team-Data-Management-System

Abkürzungsverzeichnis

UML	Unified Modeling Language
URI	Uniform Resource Identifier
VDE	Verband der Elektrotechnik, Elektronik und Informationstechnik
VDI	Verein Deutscher Ingenieure
VMT	Virtual Machine Tool
VMTL	Virtual Machine Tool Language
W3C	World Wide Web Consortium
WWW	World Wide Web
XMI	XML Metadata Interchange
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language
XSLT	XSL Transformation
YAT	Yet Another Tree-based system

1 Einleitung

1.1 Überblick

Durch die zunehmende Globalisierung der Wirtschaft sehen sich die Unternehmen den Herausforderungen internationaler Märkte gegenüber. Die Segmentierung und Sättigung der Absatzmärkte sowie stetig steigende Kundenanforderungen [LINDEMANN ET AL. 2003] führen zu einem immer größer werdenden Wettbewerbsdruck. Um sich im Umfeld dynamischer Märkte behaupten zu können, werden von den Unternehmen neue Eigenschaften gefordert, welche häufig mit den Begriffen "Agilität", "Reaktionsschnelligkeit" und "Wandlungsfähigkeit" beschrieben werden [DÜRRSCHMIDT 2001]. Vor allen Dingen kennzeichnen diese Begriffe die Stellung des Kunden in der unternehmerischen Wertschöpfungskette. Seine Rolle wandelt sich vom Wertschöpfungsempfänger zum Wertschöpfungspartner [REICHWALD & PILLER 2002] [WEBER 2004]. Die Folge sind individualisierte Produkte, die auf die Anforderungen des Kunden zugeschnitten sind und die Marktposition der Unternehmen durch ausgeprägte Alleinstellungsmerkmale festigen. Dies führt zu einem hohen Innovationsdruck und erfordert eine reaktionsschnelle Adaption der Produktionsstrukturen an die äußeren Anforderungen, um erfolgreich bleiben zu können [REINHART & CISEK 2003]. Die Unternehmen nutzen hierbei den rasanten technologischen Fortschritt, um die Produkte flexibel und individuell dem Bedarf des Kunden anzupassen. Dabei nimmt indes die Produktkomplexität stetig zu, wodurch die Entwicklungskosten und der Entwicklungsaufwand steigen, so dass sich die Schere zwischen Produktlebens- und Produktentstehungszeit immer weiter öffnet. Maßnahmen, die auf eine Wettbewerbssteigerung der Unternehmen zielen, sind nur dann erfolgreich und sinnvoll, wenn diese auf die Produktentwicklung ausgerichtet sind [VARNHAGEN 2000].

Auch traditionelle deutsche Industriezweige wie der Werkzeugmaschinenbau sind von den aufgezeigten Entwicklungen betroffen. Als wichtiger Zulieferer der Automobilindustrie nimmt der Werkzeugmaschinenbau eine Schlüsselstellung in der Leistungsfähigkeit der deutschen Wirtschaft ein. Die Werkzeugmaschinenindustrie verfügt dabei über hoch qualifizierte Experten, die durch ihre langjährige Erfahrung die internationale Wettbewerbsfähigkeit garantieren. Neue Denkansätze wie die Mechatronik zeigen jedoch, dass eine Verlagerung bzw. Erweiterung von fachspezifischen Kernkompetenzen innerhalb der Unternehmen notwendig ist, um weiterhin mit innovativen Produkten den Kundenwünschen gerecht zu werden. So ergab eine Studie im Rahmen der *Werkzeugmaschinen-Initiative 20XX* [WZM-20XX 2005], dass Werkzeugmaschinen in Zukunft mit Funktionen zur Selbstüberwachung, Fehlervorhersage und Teleservice ausgestattet sein werden. Die Bedeutung mikro-elektromechanischer Komponenten, beispielsweise für die Realisierung aktiver Werkstückspannsysteme, steigt ebenso wie der Einsatz so genannter „plug and produce“ Fertigungssysteme, bei denen aus einfachen Maschinenmodulen komplexe Systeme konfiguriert werden können.

Die steigenden Marktanforderungen bezüglich Mengenleistung, Arbeitsgenauigkeit, Flexibilität und Verfügbarkeit führen zu immer leistungsfähigeren aber auch komplexeren Maschinen und als Folge davon zu steigendem Entwicklungsaufwand. Vor diesem Hintergrund wächst das Interesse der Industrie an effizienten Arbeitsabläufen in der Entwicklung. Rationalisierungspotenzial ist vor allem im Abstimmungsbedarf zwischen den Fachbereichen gegeben.

Demzufolge sind neue Vorgehensweisen und Beschreibungsmittel notwendig, um die interdisziplinären Abstimmungsprozesse zu verbessern und ein gemeinsames, fachbereichsübergreifendes Problemverständnis zu fördern.

1.2 Ausgangssituation

Die enge Verzahnung der einzelnen Fachdisziplinen in der technischen Umsetzung spiegelt sich derzeit auf der organisatorischen Ebene noch nicht in ausreichendem Maße wider [SCHERNIKAU 2001]. Bei der Entwicklung mechatronischer Systeme tritt dieser Umstand besonders deutlich hervor, da die Kombination verschiedener Fachbereiche Produktmerkmale erlaubt, die durch eine Fachdisziplin allein nicht möglich wären [HUANG 2002].

Moderne Werkzeugmaschinen sind eine spezielle Ausprägung komplexer, mechatronischer Systeme [REINHART ET AL. 1999]. Die Wechselwirkungen zwischen den einzelnen Baugruppen werden mit den verwendeten Beschreibungsmitteln und Vorgehensweisen jedoch nur unzureichend abgebildet [EHRENSTRASSER ET AL. 2003] [KLEINER 2003]. Bei der Abstimmung der einzelnen Arbeitsschritte kann es deshalb zu Missverständnissen zwischen den Fachbereichen und Fehlern in der Systementwicklung kommen. Dies kann zu einem erhöhten zeitlichen und finanziellen Aufwand führen und die Qualität des entwickelten Produkts beeinflussen. Im Rahmen eines am Institut für Werkzeugmaschinen und Betriebswissenschaften der TU München (*iwb*) durchgeführten Workshops wurden zentrale Problemfelder im derzeitigen Entwicklungsablauf aufgezeigt [VDW 2003]. Zusammen mit den in [PÜHL 1999] angeführten Aspekten lassen sich die folgenden Randbedingungen festhalten:

- Die Mechanikkonstruktion ist die dominierende Disziplin in der Konzeptphase.
- Funktionsbeschreibungen und Prinzipskizzen werden von der Mechanikkonstruktion in der Regel nicht erstellt.
- Die Definition von Systemanforderungen wird nicht unterstützt oder ist aufgrund fehlender Informationen nicht möglich.
- Die räumliche Trennung der einzelnen Fachbereiche erschwert häufig den Informationsaustausch.
- Begriffe und Bezeichnungen werden vielfach unterschiedlich verwendet.
- Das Verständnis der Auswirkungen eigener Arbeitsschritte auf andere Fachbereiche ist unzureichend ausgeprägt.
- Ein fehlendes Änderungswesen führt zu kostenintensiven Fehlentwicklungen.

Im Werkzeugmaschinenbau hat die Mechatronik zu neuen, innovativen Konzepten und Baugruppen geführt. Gerade im Bereich der Steuerungs- und Installationstechnik ist in den letzten Jahren ein signifikanter technischer Fortschritt zu verzeichnen. So ist der Softwareanteil am Wert der Maschine in den letzten 30 Jahren kontinuierlich gestiegen, wodurch der reine Mechanikanteil auf 40% gesunken ist [BENDER 1999]. Demzufolge hat sich die Produktstruktur moderner Werkzeugmaschinen bedeutend verändert. Den steigenden Ansprüchen hinsichtlich

Dynamik und Genauigkeit kann heute durch neue Antriebskonzepte und leistungsfähigere Steuerungen begegnet werden. Ursprünglich rein mechanische Funktionsgruppen, wie beispielsweise Werkzeugwechsler, erlauben durch elektronische und softwaretechnische Unterstützung Wechselzeiten von unter zwei Sekunden. Trotz dieser Tatsache ist die Mechanikkonstruktion der dominierende Fachbereich, der die Rahmenbedingungen für alle weiteren Arbeitsschritte bestimmt. Vor allem in der Konzeptphase werden weitere Fachbereiche, wie die Elektrokonstruktion und insbesondere die Softwareentwicklung, in vielen Fällen nicht miteinbezogen. Die Steuerungssoftware wird häufig erst in der Vorinbetriebnahme am realen Hardwareprototyp getestet und an das Maschinenverhalten angepasst, was unter Umständen nur in mehreren Iterationsschleifen möglich ist und aufwändige Hardwareanpassungen erfordert.

Erschwert wird diese Arbeit durch fehlende Funktionsbeschreibungen und Prinzipskizzen vonseiten der Mechanikkonstruktion. Der Elektrokonstruktion und Softwareentwicklung stehen somit häufig nur das 2D- bzw. 3D-CAD-Modell der Werkzeugmaschine sowie mündliche bzw. formlose textuelle Beschreibungen zur Analyse von Bewegungsabläufen und Maschinenzuständen zur Verfügung. Für die angeführten Fachbereiche ist es schwierig, aus diesen Dokumenten und Informationen die für sie notwendigen Details abzuleiten und in eigenen, fachspezifischen Modellen weiter auszuführen. Die Beschreibung präziser Systemanforderungen für die Planung der weiteren Arbeitsschritte sowie für die Entwicklung und Konstruktion zentraler Systemkomponenten ist dadurch nur bedingt möglich.

Die in vielen Unternehmen anzutreffende räumliche Trennung der einzelnen Fachbereiche wird ebenfalls als Kritikpunkt bei der Weitergabe von Informationen angegeben, wobei sich hierbei ein prinzipielles Problem bei interdisziplinären Entwicklungsprozessen zeigt. Die einzelnen Fachbereiche arbeiten in einer ihrem Arbeitsschwerpunkt entsprechenden Begriffswelt, mit spezifischen Beschreibungs- und Strukturierungsmitteln sowie Notationselementen. Den Begriff *Komponente* setzt der Mechanikkonstrukteur beispielsweise mit einem Bauteil oder einer Baugruppe gleich. Eine nähere Beschreibung erfolgt im CAD-Modell. Für den Softwareentwickler wird damit jedoch der Softwareanteil eines Bauteils oder einer Baugruppe bezeichnet. Die Beschreibung dieser Softwarekomponente erfolgt in einem geeigneten Modell zur Abbildung von Softwarestrukturen. Ein durchgängiger Informationsfluss sowie eine gesamtheitliche Modellierung der Maschineneigenschaften ist derzeit nicht möglich und führt letztendlich dazu, dass die fachbereichsübergreifenden Auswirkungen von Arbeitsergebnissen auf nachfolgende Arbeitsschritte nur unzureichend berücksichtigt werden. Darüber hinaus fehlt in vielen Unternehmen ein Änderungswesen, welches über die Grenzen der einzelnen Fachbereiche hinweg Änderungen protokolliert und kostenintensive Fehlentwicklungen zu verhindern versucht.

Um dem mechatronischen Charakter bei der Produktentwicklung gerecht zu werden, ist eine frühzeitige Einbindung und Abstimmung aller Fachbereiche erforderlich [REINHART ET AL. 2002A]. In zahlreichen Arbeiten wird deshalb vorgeschlagen, aus einer gemeinsamen, interdisziplinären Sicht zunächst die Maschinenfunktionen schrittweise zu spezifizieren und sie schließlich in den einzelnen Fachbereichen weiter zu detaillieren. Ein derartiges Funktionsmodell ist die Basis aller weiteren fachspezifischen Arbeitsschritte und unterstützt den interdisziplinären Entwurf mechatronischer Konzepte [REINHART ET AL. 2001A] [LANGLOTZ 2000] [FLATH 2002] [HUANG 2002]. Aufbauend auf dem Funktionsmodell sollen diese direkt am

Rechner simuliert und getestet werden können. Die verschiedenen Aspekte einer Werkzeugmaschine werden dabei in unterschiedlichen Dokumenten und Modellen beschrieben. Um die widerspruchsfreie Beschreibung der Maschinenfunktionalität gewährleisten zu können, ist eine dokument- und modellübergreifende Konsistenz der Entwicklungsdaten sicherzustellen.

Werden verschiedene fachbereichsspezifische Maschinenmodelle zu einem Gesamtmodell zusammengefasst und dieses für die Simulation von Maschineneigenschaften eingesetzt, so spricht man vom virtuellen Prototypen einer Maschine [REINHART ET AL. 2002B] [ENGLBERGER ET AL. 2003] [KREUSCH 2002]. Der Aufbau eines virtuellen Modells einer Werkzeugmaschine im Rechner erfordert die Verknüpfung einer Vielzahl von unterschiedlichen Softwarewerkzeugen, welche die Fachbereiche zur Abbildung einzelner Aspekte des Maschinenverhaltens einsetzen. Die datentechnische Integration dieser Softwaresysteme ist hierbei ein entscheidender Faktor, um neue Optimierungs- und Leistungspotenziale für die Werkzeugmaschinenentwicklung erschließen zu können [WECK & POSSEL-DÖLKEN 2003].

1.3 Motivation und Zielsetzung

Virtuelle Produktentwicklungsmethoden erlauben Zeit und Kosten zu sparen, allerdings nur dann, wenn die dafür notwendigen Abläufe beherrscht werden. Der Einsatz neuer Vorgehensmodelle zur verbesserten Einbindung der einzelnen Fachbereiche in den Entwicklungsprozess ist hierbei ein entscheidender Baustein. Die Aufgabe besteht darin, über alle Fachbereichsgrenzen hinweg ein gemeinsames Problemverständnis zu schaffen, das eine gezielte interdisziplinäre Abstimmung der einzelnen Arbeitsschritte ermöglicht. Insbesondere die Erstellung der Steuerungssoftware wird derzeit in der Werkzeugmaschinenentwicklung noch nicht aktiv mit einbezogen. Die Mechanikkonstruktion dominiert meist die Arbeitsabläufe und bestimmt dabei die Anforderungen, Meilensteine und Rahmenbedingungen. Die Software als vermeintlich flexibelster Baustein muss sich dann den Gegebenheiten im Gesamtprojekt anpassen [GLATZ 2000]. Arbeiten aus dem universitären Bereich sowie der Werkzeugmaschinenindustrie zeigen, dass vor allem der verstärkte Einsatz von Simulationsmodellen den Abstimmungsprozess zwischen der Mechanikkonstruktion und der Softwareentwicklung verbessert. Anhand von 3D-Simulationen können erste Funktionsmuster direkt am Rechner betrachtet und Bewegungsabläufe interaktiv analysiert und weiterentwickelt werden.

Um eine zeit- und kostenoptimale Erstellung dieser Simulationsmodelle zu ermöglichen, sind neue, interdisziplinäre Modellierungstechniken erforderlich, die eine durchgängige Beschreibung der Maschineneigenschaften von der Anforderungsanalyse bis hin zur Inbetriebnahme erlauben. Wesentliche Reibungsverluste entstehen hier bei der Integration der verschiedenen Beschreibungstechniken bzw. zwischen den Softwarewerkzeugen der einzelnen Fachbereiche. Benötigt wird eine durchgängige und fachbereichsübergreifende Softwareunterstützung für die gesamte Maschinenentwicklung, welche die verschiedenen Beschreibungstechniken zusammenführt und eine fachbereichsübergreifende Gesamtsicht auf die Maschine ermöglicht.

In dieser Arbeit wird das Konzept eines Softwaresystems vorgestellt, das die Integration der Daten von verschiedenen, fachbereichsspezifischen Softwarewerkzeugen zum Ziel hat. Die Integration basiert auf einem Datenmodell, das die zentralen Beschreibungselemente eines jeden Fachbereichs abbildet und sie in einem gemeinsamen Kontext zusammenführt (siehe Abbildung 1-1).

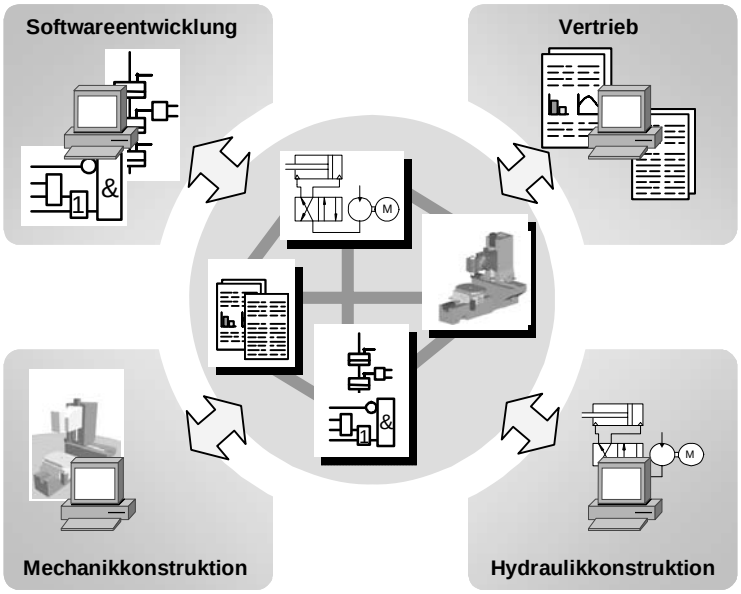


Abbildung 1-1: Integration der einzelnen Fachbereiche in einen gemeinsamen Kontext

Bei der Entwicklung von Werkzeugmaschinen entsteht eine Fülle auf komplexe Weise miteinander in Verbindung stehender Dokumente und Modelle [RUMPE 1996]. Diese Verbindungen werden in dem Datenmodell abgebildet und definieren den gemeinsamen Kontext in der Maschinenentwicklung. Das Datenmodell kann somit als interdisziplinäres Glossar angesehen werden, in dem die verwendeten fachspezifischen Begriffe und Beschreibungselemente in Relation zueinander gesetzt werden. In Bezug auf die in Abschnitt 1.2 aufgeführten Defizite wird dadurch der Informationsfluss zwischen den Fachbereichen verbessert, ein einheitliches Begriffs- und Problemverständnis geschaffen sowie die Transparenz technischer Zusammenhänge erhöht. Das Softwaresystem, im Folgenden als Metadaten-Management-System bezeichnet, verwaltet das Datenmodell. Es besitzt eine Client-Server-Architektur und ist Teil eines integrierten, modellgetriebenen Entwicklungsarbeitsplatzes zum Aufbau virtueller Prototypen. Die Systemarchitektur und die Funktionsweise des Metadaten-Management-Systems sowie dessen Einsatz im integrierten Entwicklungsarbeitsplatz bilden den Schwerpunkt dieser Arbeit.

1.4 Vorgehensweise

Ausgehend von der in Kapitel 1 geschilderten Aufgabenstellung und Zielsetzung gehen die folgenden Kapitel auf den Stand der Technik im Werkzeugmaschinenbau ein, erläutern das Konzept zur Erreichung der vorgestellten Zielsetzung und demonstrieren die erarbeitete Lösung anhand von Anwendungsbeispielen (siehe Abbildung 1-2).

Die für die Beschreibung und Integration von Daten notwendigen Grundlagen werden in Kapitel 2 erläutert und im Anhang weiter ausgeführt (siehe Kapitel 11).

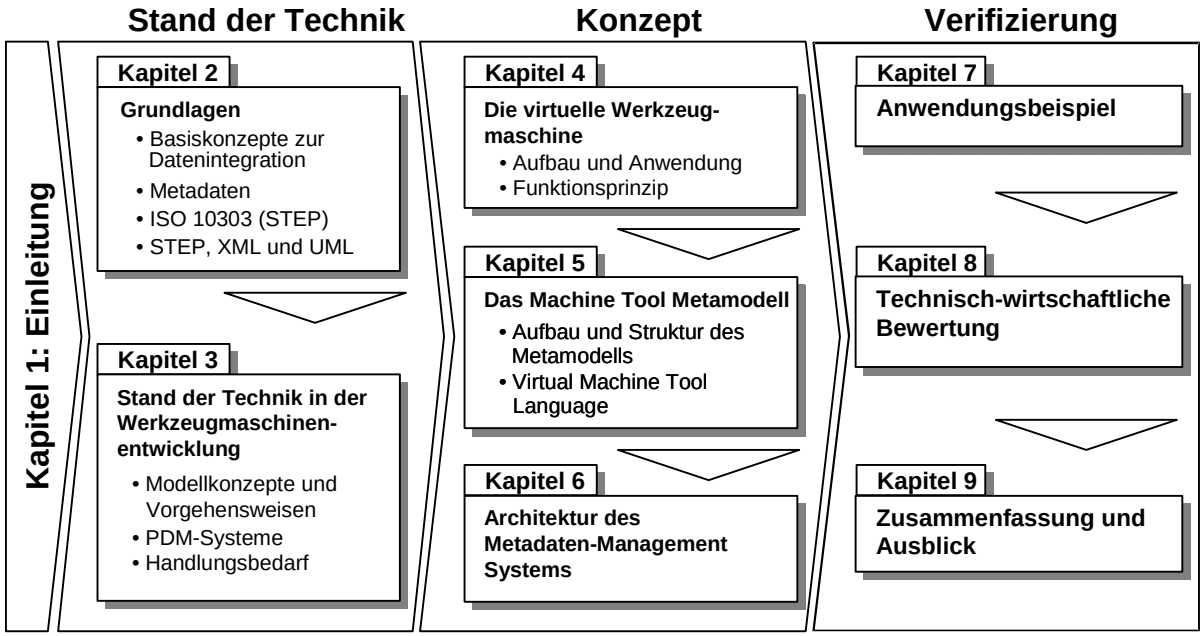


Abbildung 1-2: Aufbau der Arbeit

In Kapitel 3 wird der Stand der Technik in der Werkzeugmaschinenentwicklung zusammengefasst und diskutiert. Insbesondere werden Modellkonzepte und Vorgehensweisen sowie Entwicklungsumgebungen betrachtet und bewertet. Von zentralem Interesse sind hierbei der Aufbau der eingesetzten Modellierungssprachen und -techniken sowie deren Anwendung in entsprechenden Softwarewerkzeugen. Die Diskussion und Einordnung von Produktdaten-Management-Systemen sowie der Technologien in deren Umfeld gibt abschließend einen Überblick, wie weit die Integration von heterogenen Daten unterschiedlicher Fachbereiche fortgeschritten ist. Die Diskussion und Bewertung der dargestellten Techniken und Konzepte dient als Grundlage für die abschließende Formulierung des konkreten Handlungsbedarfs.

In Kapitel 4 wird das in dieser Arbeit entwickelte Konzept der virtuellen Werkzeugmaschine als integrierter Entwicklungsarbeitsplatz vorgestellt und dessen Funktionsprinzip formal hergeleitet. Die Darstellung einer konkreten Ausprägung dieses theoretischen Modells erfolgt in Kapitel 5. Darüber hinaus wird hier eine Methodik zum Aufbau der virtuellen Werkzeugmaschine als Informationsverbund verschiedener Softwarewerkzeuge definiert. Anschließend wird in Kapitel 6 die Systemarchitektur der wesentlichen Kernkomponenten zur Datenintegration der virtuellen Werkzeugmaschine beschrieben.

Kapitel 7 dokumentiert die praktische Umsetzung des vorgestellten Konzeptes anhand zweier Anwendungsfälle im industriellen Umfeld. In Kapitel 8 erfolgt die Bewertung des erarbeiteten Lösungsansatzes aus technischer und wirtschaftlicher Sicht. Die Arbeit schließt in Kapitel 9 mit einer Zusammenfassung und einem Ausblick auf zukünftige Einsatzmöglichkeiten sowie Anwendungsszenarien der dargelegten Ergebnisse im Hinblick auf einen integrierten Entwicklungsarbeitsplatz.

Der Anhang in Kapitel 11 enthält Begriffsdefinitionen, Ausführungen zu grundlegenden Technologien sowie eine Auflistung der in dieser Arbeit eingesetzten Softwarewerkzeuge. Abschließend werden die deutschen Übersetzungen englischer Zitate tabellarisch aufgeführt.

2 Grundlagen

2.1 Übersicht

In diesem Kapitel werden grundlegende Konzepte für die Beschreibung und Integration von Daten erläutert. Insbesondere werden Konzepte und Techniken vorgestellt, um die Strukturen von Daten sowie deren Semantik beschreiben zu können.

2.2 Basiskonzepte zur Datenintegration

In den global operierenden Unternehmen liegen Daten häufig lokal in den Zweigstellen vor und müssen erst zu brauchbaren Informationen zusammengeführt und transformiert werden. Dies ist mit großem Aufwand verbunden, da sich diese Daten in ihrer Syntax und Semantik unterscheiden. Um mit solchen verteilten, heterogenen Datenquellen arbeiten zu können, sind geeignete Methoden zur Datenintegration erforderlich. Dabei geht es nicht nur darum, Daten in einer bestimmten Art und Weise zusammenzufügen, sondern auch darum, wie diese weiterverarbeitet und wie Wissen abgeleitet werden kann, um fundierte Entscheidungsprozesse initiieren zu können. Im Folgenden werden zentrale Konzepte für die Integration von Daten erläutert, auf welche das in Kapitel 4 vorgestellte System aufbaut.

2.2.1 Semistrukturierte Daten und Datenmodelle

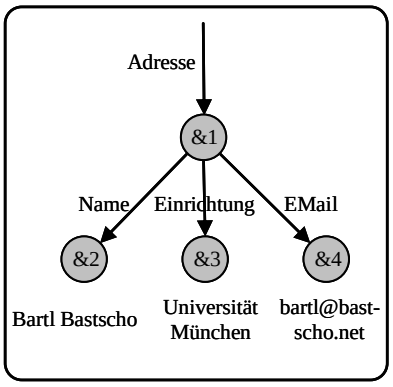
Traditionelle Datenbanksysteme setzen voraus, dass die gespeicherten Daten nach einem im Voraus festgelegten Schema strukturiert sind. In einer dezentral verwalteten Datenbasis wie dem Internet ist ein derartiges Schema oft zu restriktiv, da die einzelnen Daten in unterschiedlichen, heterogenen Formaten vorliegen [BRY ET AL. 2001]. Einige dieser Daten werden in relationalen oder objekt-orientierten Datenbanken abgespeichert, wodurch deren Struktur klar festgelegt ist. Auf der anderen Seite existieren Daten, die unstrukturiert vorliegen, wie etwa Bilder oder unformatierter Text. Die meisten Datenquellen befinden sich jedoch zwischen diesen beiden Extremen [SUCIU 1997]. Formatierte Texte, wie beispielsweise HTML-Dokumente, weisen durch Überschriften und Absätze eine implizite Struktur auf. Um die in ihnen enthaltenen Informationen nutzen zu können, muss deren Struktur zuvor extrahiert werden. Da derartige Dokumente nach einer bestimmten Grammatik aufgebaut sind, können Parser diese Aufgabe übernehmen. Dokumente, die zusätzlich noch Bilder oder unformatierten Text enthalten, weisen lediglich eine partielle Struktur auf. Daten, die diese Eigenschaften besitzen, werden als semistrukturierte Daten bezeichnet [BUNEMAN ET AL. 1997].

Für die Verarbeitung semistrukturierter Daten in Informationssystemen ist es erforderlich, eine gemeinsame, darunter liegende Struktur zu ermitteln. Diese bildet die Grundlage für eine Anfrageauswertung von Seiten des Anwenders ebenso wie für eine effiziente Datenverwaltung innerhalb des Informationssystems [NESTRorov ET AL. 1997]. Auf Grund der Unregelmäßigkeit ist eine Beschreibung semistrukturierter Daten in relationalen oder objekt-orientierten Datenmodellen nicht möglich. So lassen sich tief verschachtelte Daten nur unzureichend in den flachen Tabellen relationaler Datenmodelle ablegen. Dies ist zwar mit objekt-orientierten Datenmodellen möglich, allerdings lassen sich hier keine unregelmäßigen Daten-

strukturen mit fehlenden oder sich wiederholenden Teilstrukturen repräsentieren [SUCIU 1998]. Diese Sachverhalte lassen sich in einem Beispiel einfach vor Augen führen. Abbildung 2-1 a zeigt einen Ausschnitt einer Adresskartei in der Sprache *Extensible Markup Language* (XML, siehe Anhang 11.3.1). Es sind zwei Adressen als Datensätze enthalten. Der zweite besitzt eine reichere Struktur als der erste, des Weiteren befindet sich die E-Mail-Adresse in beiden Datensätzen nicht in derselben Tiefe. Mit der XML lassen sich derartige Datenstrukturen spezifizieren. Anders als etwa in relationalen Datenmodellen wird in der XML die Struktur in den Datensätzen selbst wiedergegeben, man spricht auch von strukturtragenden oder selbsterklärenden Daten. Dadurch ist die Abbildung alternativer Strukturen wie in Abbildung 2-1 a möglich. Die XML ist somit als Datenmodellierungssprache für semistrukturierte Daten geeignet.

```
<Adresskartei>
  <Adresse>
    <Name>Bartl Bastscho</Name>
    <Einrichtung>Universität München</Einrichtung>
    <Email>bartl@bastscho.net</Email>
  </Adresse>
  <Adresse>
    <Name>Susi Bastscho</Name>
    <Einrichtung>Univesität München</Einrichtung>
    <Abteilung>Fakultät für Mathematik und Informatik</Abteilung>
    <Kontakt>
      <Email>susi@bastscho.net</Email>
      <URL>http://www.bastscho.net/freunde/susi</URL>
    </Kontakt>
  </Adresse>
</Adresskartei>
```

a) Adresskartei in XML



b) OEM-Objekt Adresse

Abbildung 2-1: Adresskartei in XML- (a) und OEM- (b) Darstellung [BRY ET AL. 2001]

Allgemein haben sich baumartige Graphen als bevorzugtes Beschreibungsmittel zur Modellierung semistrukturierter Daten durchgesetzt. Das *Object Exchange Model* (OEM) [PAPAKONSTANTINOU ET AL. 1995] ist der bekannteste Vertreter solcher semistrukturierter Datenmodelle (siehe Abbildung 2-1 b). In einem OEM-Modell werden Datensätze als eine Menge von Objekten beschrieben, wobei jedes Objekt atomar oder komplex sein kann. Ein OEM-Objekt lässt sich als Tupel in der Form *(attribute, identifier, value)* darstellen. *Attribute* sind Strings aus einer Menge von Attributnamen und bezeichnen Begriffe, mittels derer auf die einzelnen OEM-Objekte zugegriffen werden kann. *Identifier* dienen dagegen der eindeutigen Objekt-Identifikation. Das Tupel-Element *value* hat je nachdem, um welche Art von OEM-Objekt es sich handelt, eine unterschiedliche Ausprägung. Im Falle atomarer Objekte ist *value* ein Element aus einer Menge skalarer Basis-Typen wie Integer oder String, bei komplexen Objekten besteht *value* aus einer Menge von (Sub-)Objekten.

Formal lässt sich das OEM-Modell mit dem Tupel $G=(V,E,r,v)$ darstellen. Die Menge $V=V_c \cup V_a$ setzt sich hierbei aus komplexen und atomaren Knoten zusammen, die Kanten E werden als kartesisches Produkt $E \subseteq V_c \times A \times V$ abgebildet, wobei A die Menge der Kantenbezeichnungen darstellt. $r \in V$ ist der Wurzel-Knoten, von dem aus alle Knoten eines OEM-Objektes erreichbar sind. Die Abbildung $v: V_a \rightarrow D$ weist dem atomaren Knoten Elemente aus der Menge der atomaren Werte D zu [SUCIU 1998]. Ein OEM-Objekt kann als gerichteter Multigraph [STEGER 2001] angesehen werden, wodurch die Definition zyklischer Objekte

zulässig ist. Die Knoten sind eindeutige Objektbezeichner und werden mit einem einleitenden &-Zeichen versehen, während die Kanten die Attribute des jeweiligen OEM-Objektes darstellen. Anfragen nach bestimmten Daten werden durch das Durchlaufen einzelner Kanten im OEM-Graph verarbeitet. Sind beispielsweise alle Adressen der Adresskartei zurückzugeben, so werden die Kanten mit dem Attribut *Adresse* durchlaufen. Soll die Adresse einer bestimmten Person gesucht werden, so muss die Kante mit den beiden Attributen *Adresse.Name* verfolgt werden. Suchanfragen können unter der Berücksichtigung nur partiell bekannter Strukturen auch mit regulären Ausdrücken erfolgen, so liefert der Ausdruck **.EMail* alle E-Mail-Adressen zurück, die im OEM-Graph enthalten sind. Die ausführliche Beschreibung der Datenabfragesprache *OEM-QL* kann in [PAPAKONSTANTINOU ET AL. 1995] nachgeschlagen werden.

Das *Yet Another Tree-based System* (YAT) ist ein weiteres Beschreibungsmittel, um semistrukturierte Daten abzubilden [Cluet et al. 1998]. Ebenso wie OEM verwendet auch YAT Baum-Graphen zur Darstellung von Datenstrukturen. Ein YAT-Datenmodell setzt sich aus so genannten Patterns zusammen, wobei ein Pattern aus einer Menge von geordneten Bäumen besteht. Diese werden innerhalb eines Patterns mit Oder-Operatoren verknüpft dargestellt. Die Knoten eines jeden Baumes beschreiben Variablen oder Konstanten. In YAT können Variablen werte- bzw. modellbezogen sein und entsprechend zur Speicherung von Daten oder zur Referenzierung von Patterns verwendet werden. Das Referenzieren von Patterns durch eine Variable ermöglicht es, bereits existierende Patterns zu instanziieren.

In Abbildung 2-2 a sind die grundlegenden Konstrukte von YAT dargestellt. Für die Instanziierung dieses Patterns muss einer der drei Bäume ausgewählt und mit entsprechenden Werten belegt werden. Die Variable *L* stellt hierbei eine Datenvariable dar. Diese werden in YAT immer mit einem Großbuchstaben gekennzeichnet. Variablen mit Modellbezug, *Pattern-Variablen* genannt, werden im Fettdruck abgebildet. Mit ihnen lassen sich komplexe, rekursive Baumstrukturen abbilden. Die Kanten werden mit Indikatoren versehen, welche die Häufigkeit eines Knotens angeben. Das Symbol "*" bezeichnet hierbei eine Sequenz von null bis unendlich viele Knoten. Kanten ohne eine Beschriftung geben an, dass ein Knoten exakt einmal existiert. Solche Kanten werden auch als einfache *Kanten* bezeichnet. Pattern-Variablen mit einem einleitenden Zeichen „&“ stellen eine Referenz zu einem anderen Pattern her.

Das Konzept der Instanziierung lässt sich anhand des Beispiels in Abbildung 2-2 b verdeutlichen. Das hier dargestellte YAT-Modell definiert den Aufbau einer Klasse aus der objektorientierten Programmierung.

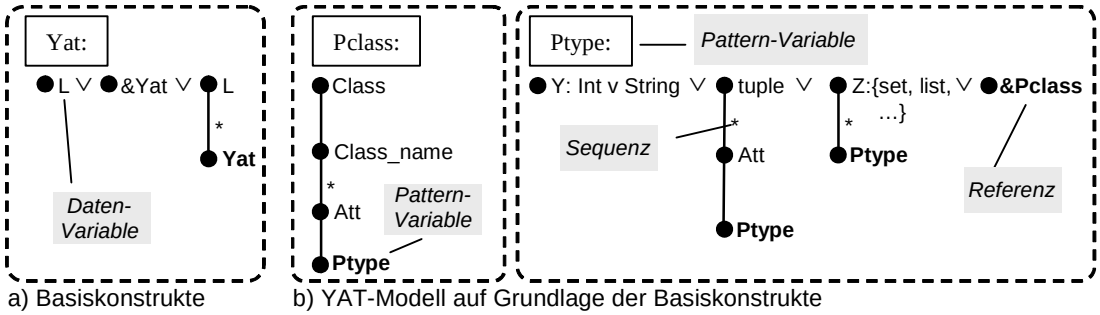


Abbildung 2-2: Das YAT-Datenmodell

Eine Klasse wird mit einem Namen (*Class_name*) gekennzeichnet und besitzt eine Menge von *Attributen (Att)* eines bestimmten *Typs (Ptype)*. In Bezug auf den letzten Baum rechts aus Abbildung 2-2 a und den Unterbaum mit Wurzel *Att* aus Abbildung 2-2 b lässt sich das Pattern *Pclass* als Instanz von YAT erkennen. Dasselbe gilt für den Unterbaum mit der Wurzel *Class_name*.

Das Konzept der Instanziierung in YAT basiert auf den Kantenbeschriftungen. Eine Kante ohne Beschriftung (einfache Kante) kann lediglich durch eine einzelne Kante ersetzt werden. Eine Kante mit der Beschriftung "*" steht stellvertretend für eine Sequenz von Kanten, die selbst eine Beschriftung besitzen können.

Durch das Konzept der Modellreferenzen ist die Darstellung eines Modells über mehrere Instanziierungsebenen möglich. Das Pattern *Pcar* aus Abbildung 2-3 a ist beispielsweise eine Instanz von *Pclass* (siehe Abbildung 2-2). Die Variable *Class_name* ist durch die Variable *car* instanziiert und die Variable *Att* durch die Variable *name*. Ebenso ist das Pattern *c1* aus Abbildung 2-3 b eine Instanz von *Pclass*. Die Variable car ist eine Instanz der Variable *Class_name*. Die Menge der ausgehenden Kanten von *car* entspricht dem Unterbaum mit der Wurzel *Class_name* vom Pattern *Pclass*.

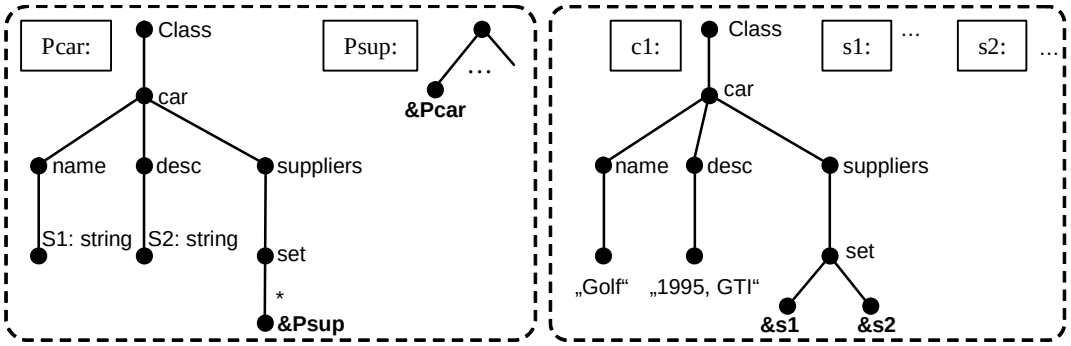


Abbildung 2-3: Instanziierung von YAT-Datenmodellen

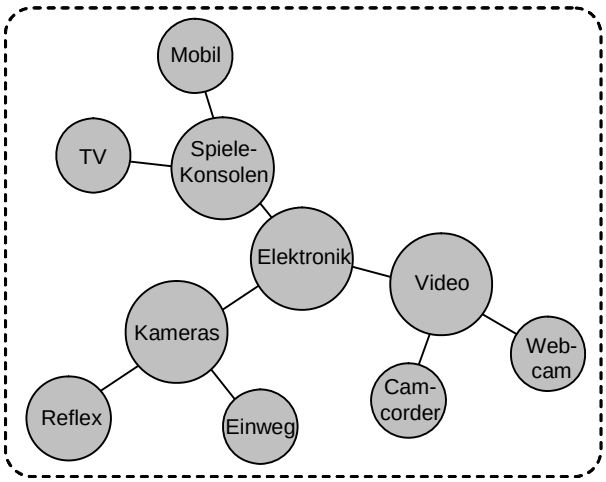
Die ausgehende Kante der Variablen *name* aus dem Pattern *c1* endet in dem Knoten mit der Bezeichnung *Golf*. *Golf* ist eine Instanz von *Ptype*, wie aus der Definiton der Datenvariable *Y* aus Abbildung 2-2 b ersichtlich ist. Die Variable *Y* bezeichnet einen String- oder Integer-Wert.

Besteht ein Pattern aus nur einem Baum und ist dieser lediglich aus einfachen Kanten aufgebaut, so wird dieses Pattern als *Ground Pattern* bezeichnet (siehe *c1* aus Abbildung 2-3 b). Ground Pattern sind von ihrem Abstraktionsgrad mit den OEM-Graphen zu vergleichen. Es existieren noch weitere Ansätze, um semistrukturierte Daten abzubilden. Einen guten Überblick geben die Arbeiten von [Suciu 1997], [Suciu 1998] und [Buneman et al. 1997] sowie [Cardelli 2001] und [Nestrorov et al. 1997].

2.2.2 Ontologie

In vielen Bereichen steht man vor der Aufgabe, Wissen zu kommunizieren und zu repräsentieren, zum Beispiel über Fakten, Sachverhalte oder Regeln in einem technischen Anwendungsbereich, einem Geschäftsprozess oder über Inhalte von Dokumenten oder Webseiten. Der Mensch macht sich Wissen zunutze, indem er auf seine Erfahrung und auf sein Grundlagen- und Kontextwissen zurückgreift, das er mit den Inhalten aus Lexika, Lehrbüchern oder Schlagwortregistern in Verbindung setzt [HESSE 2002]. Soll gespeichertes Wissen maschinell von Informationssystemen verarbeitet werden, ist eine rechnerinterpretierbare, algorithmisch erfassbare Repräsentation der zu Grunde liegenden Begriffe und deren Zusammenhänge erforderlich. Hierfür können ontologische Modelle eingesetzt werden.

Die Ontologie ist ursprünglich eine philosophische Disziplin, die sich primär mit dem Sein bzw. mit den Möglichkeiten und Bedingungen des „Seienden" beschäftigt und ist somit eng verwandt mit der Erkenntnistheorie, die sich mit dem menschlichen Wahrnehmen und Erkennen beschäftigt [WEISSMAHR 1991]. In der Informatik wird darunter ein formal definiertes System mit dem Ziel eines strukturierten und fundierten Aufbaus von Wissensbasen verstanden. In [GRUBER 1993] wird eine Ontologie als *"...explicit specification of a conceptualization"* bezeichnet. Sie wird als formales Modell eines Anwendungsbereiches definiert, das den Austausch und das Teilen von Wissen erleichtert. Eine Ontologie besteht demnach aus einer präzise festgelegten Terminologie sowie aus Beziehungen und Regeln zwischen den dort definierten Begriffen (siehe Abbildung 2-4).



Ontologien…

- **beschreiben Begriffe innerhalb eines Wissensbereichs, auf die sich eine Gruppe von Anwendern geeinigt hat.**
- **benutzen Klassifizierungen, um diese Begriffe zu gliedern und zu strukturieren.**
- **erlauben die Definition von Beziehungen und Regeln zwischen den Begriffen.**
- **dienen der Navigation in Wissensbasen.**
- **werden entwickelt, um eine maschinenverarbeitbare Semantik für Datenquellen bereitzustellen.**

Abbildung 2-4: Ontologisches Modell zur Gliederung des Artikelbestands eines Elektrofachgeschäfts

Nach [TZITZIKAS ET AL. 2001] lässt sich dies mit dem Tupel $O = (T, \preceq, \sim)$ ausdrücken, wobei T die Terminologie, $\preceq$ eine reflexive, transitive Relation und $\sim$ die Äquivalenzrelation über der Menge T darstellen. Sind $a, b \in T$ und $a \preceq b$, so sagt man: „a ist b zugeordnet". Die Relationen zwischen den Begriffen lassen sich auch in Form eines Graphen darstellen. In Abbildung 2-4 ist die Terminologie T einer einfachen Ontologie O zur Einordnung des Artikelbestandes eines Elektrofachgeschäfts zu sehen. Unter den hier definierten Begriffen werden die einzelnen

Artikel eingeordnet und verwaltet, wobei hier auf die Definition weit reichender Regeln zwischen den einzelnen Begriffen verzichtet wurde. Ist dieses ontologische Modell die Grundlage eines Informationssystems, so können damit alle Daten verwaltet werden, die mit den definierten Begriffen kompatibel sind. In Abbildung 2-5 ist ein Beispielszenario zu sehen, das den Sachverhalt veranschaulicht. Zwei Elektrofachgeschäfte möchten Artikeldaten austauschen, welche in den jeweiligen Informationssystemen abgelegt sind. Der Im- und Export soll anhand von XML-Dokumenten erfolgen. Das XML-Format verwendet dabei Annotationen, welche auf den Begriffen basieren, die in den Ontologien der jeweiligen Informationssysteme definiert sind. Das Elektrofachgeschäft A verwendet die in Abbildung 2-4 abgebildete Ontologie zur Klassifizierung der Artikelbestände. Das Informationssystem des Elektrofachgeschäfts B weicht davon geringfügig ab und verwendet den Begriff *Videokamera* anstatt *Camcorder*. Obwohl die beiden unterschiedlichen Begriffe äquivalent sind, ist ein störungsfreier Datenaustausch zwischen diesen beiden Informationssystemen nicht gewährleistet, da eine entsprechende Äquivalenzrelation nicht in den beiden Ontologien enthalten ist.

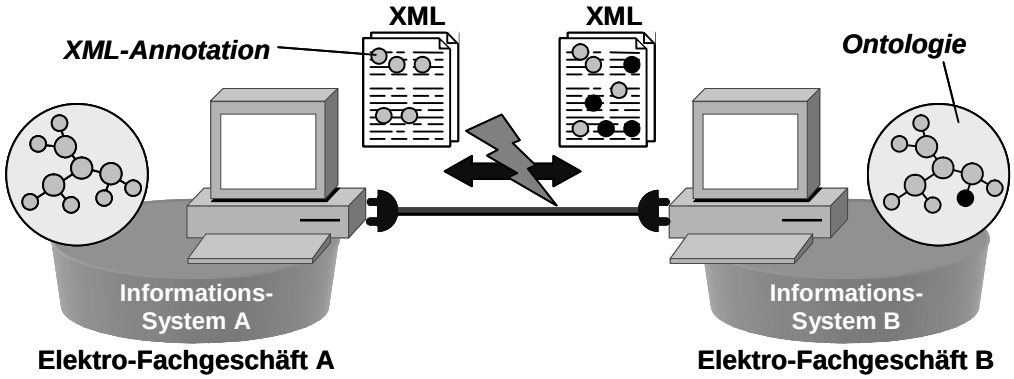


Abbildung 2-5: Austausch von Artikeldaten auf der Grundlage unterschiedlicher Ontologien

Eine Ontologie lässt sich somit auch als ein explizites Element in der Kommunikation zwischen zwei oder mehreren Akteuren verstehen, wobei es sich bei den Akteuren sowohl um Menschen als auch um technische Systeme handeln kann. Nur wenn sie alle auf dieselbe Ontologie zugreifen, können Informationen transportiert und Wissen vermittelt werden. Besonderes Interesse finden Ontologien im Rahmen der *Semantic-Web-Initiative* von Tim Berners-Lee [GRUNINGER & LEE 2002]. Das Ziel hierbei ist es, das World Wide Web (WWW) um maschinenlesbare Daten zu erweitern, welche die Semantik der Inhalte formal festlegen [BERNERS-LEE ET AL. 2001]. Beispielsweise wird die Bedeutung einzelner WWW-Links durch Annotationen klar definiert. Es kann festgelegt werden, ob ein Link auf die Homepage des Autors verweist oder zu einer Seite mit einem übergeordneten Thema führt. Die Annotationen sind in einer Ontologie definiert. Durch die Formulierung entsprechender Regeln zwischen den Annotationen kann eine Suchmaschine logische Ableitungen durchführen. Besagt die Annotation einer Webseite beispielsweise, dass sie sich mit Fußball beschäftigt und geht aus der zu Grunde liegenden Ontologie hervor, dass Fußball eine Sportart ist, so erkennt die Suchmaschine, dass auf dieser Seite das Thema Sport behandelt wird. Für Suchmaschinen wird es dadurch wesentlich einfacher, die richtigen Informationen als Antwort auf eine bestimmte Suchanfrage zu liefern.

Nähere Informationen zum Thema Semantic Web können in [DACONTA ET AL. 2003] oder [W3C 2001] nachgelesen werden.

Die Formulierung von Ontologien kann in speziellen formalen Sprachen erfolgen, den so genannten Ontologiesprachen. Ein bekannter Vertreter hiervon ist das *Resource Description Framework* (RDF) [W3C 2005]. Weitere Ontologiesprachen sind unter anderem *DARPA Agent Markup Language und Ontology Inference Layer* (DAML+OIL) [SHILING 2001], *Web Ontology Language* (OWL) [W3C 2004B] sowie das *Knowledge Interchange Format* (KIF) zum Austausch von maschinenlesbarem Wissen zwischen verschiedenen Softwaresystemen [GENESERETH & FIKES 1992].

2.2.3 Mediatoren

In [WIEDERHOLD 1992] werden Mediatoren erstmals als eine konzeptionelle Basis für die Entwicklung von Informationssystemen vorgestellt. Ein Mediator ist demnach eine Softwarekomponente, welche eine Vermittlerrolle zwischen den Anbietern von Daten (Datenquellen) und den Anwender-Applikationen (Clients) einnimmt (siehe Abbildung 2-6).

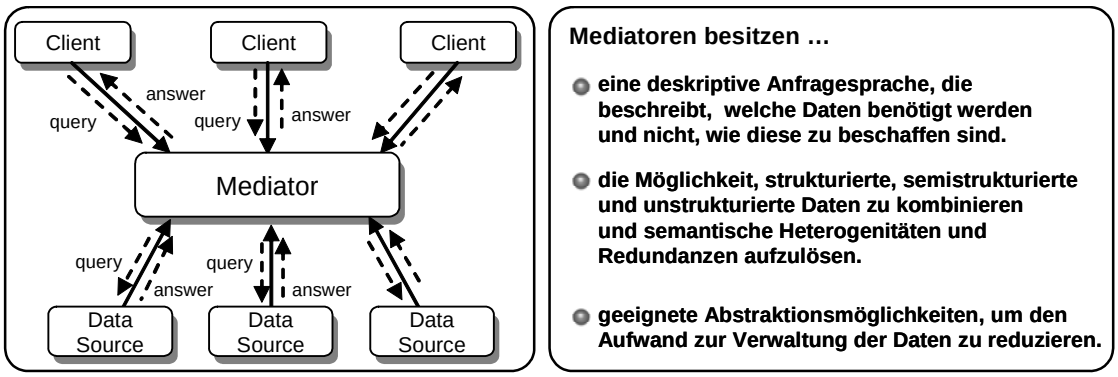


Abbildung 2-6: Architekturskizze eines Mediators

Entscheidungs- und strategierelevante Informationen sind im Unternehmen meist auf verschiedene Datenquellen verstreut. Entsprechend dem Aufgabenumfeld der einzelnen Unternehmensbereiche weisen diese Datenquellen unterschiedliche Strukturen auf. Die gezielte Abfrage nach Daten zur Informationsgewinnung ist durch die heterogene Datenverwaltung dementsprechend schwierig. Die wesentliche Aufgabe eines Mediators ist es deshalb, dem Anwender eine einheitliche Anfrageschnittstelle zu unterschiedlichen Datenquellen zur Verfügung zu stellen [DOAN & HALEVY 2005]. Der Mediator dient somit als zentraler Zugangspunkt zu den Daten und Informationen eines Informationssystems (siehe Abbildung 2-6), speichert jedoch selbst keine Daten ab. Die Vermittlungsfunktion zwischen der Anwenderebene und den einzelnen Datenquellen wird auch als *Mediation* bezeichnet. Nach [WIEDERHOLD 1992] lässt sich der Vorgang der Mediation wie folgt charakterisieren:

"Mediation simplifies abstracts, reduces, merges and explains data. […] Mediation covers a wide variety of functions that enhance stored data prior to their use in an application. Mediation makes an interface intelligent by dealing with representation and abstraction problems that you must face when trying to […] use data and knowledge resources. "

Mediatoren übernehmen eine wichtige Kapselungsfunktion, indem sie alle Zugriffe auf die Datenquellen verwalten [SCHIMKAT 2003]. Insbesondere unterstreicht [WIEDERHOLD 1992] die Fähigkeit der Mediatoren zur *Abstraktion* und *Transformation* von Daten. Das Ziel hierbei ist es, die originären Daten der Datenquellen derart aufzubereiten und zu transformieren, dass sie auf unterschiedlichen Abstraktionsstufen abgebildet werden können.

Die Daten in den einzelnen Datenquellen besitzen ein geringes Abstraktionsniveau, da sie an konkrete Attribute und Eigenschaften der jeweiligen Datenquelle gebunden sind [SCHIMKAT 2003]. Als Beispiel sei eine firmeninterne Datenbank mit Vertriebs- und Produktdaten genannt. Sie enthält konkrete Daten über die einzelnen Produkte, die Produktkosten sowie genaue Angaben über die Flugpläne der Vertriebsmitarbeiter. Die Geschäftsleitung kann nun anhand geeigneter Abstraktionen und Transformationen die für sie interessanten Informationen ableiten. Die Angaben zu den Produkten liefern die Zahl der angebotenen Produkttypen, aus den Produktkosten kann der Inflationstrend der nächsten Jahre berechnet werden und die Flugpläne der Vertriebsabteilung geben Aufschluss über die mittlere Flugdauer und die mittleren Flugkosten.

Für die Umsetzung der Kapselungsfunktion und die Fähigkeit zur Abstraktion und Transformation besitzen Mediatoren ein Datenmodell bzw. Schema, das die semantischen Abhängigkeiten zwischen dem Mediator und den jeweiligen Datenquellen abbildet. Das Schema ermöglicht es, dass Datenanfragen an die geeignete Datenquelle gestellt und an die Anwender-Applikationen in der entsprechenden Abstraktion weitergeleitet werden können. Es ist somit die Grundlage für die vom Mediator zur Verfügung gestellte Vermittlungsfunktion. Um die aufgezeigten Unterschiede bei der Namensgebung oder Granularität der Daten im Schema des Mediators zusammenzuführen, sind geeignete Techniken erforderlich.

In [TZITZIKAS ET AL. 2001] wird das Modell eines Mediators erläutert, das zur Definition des zentralen Schemas das Konzept der Ontologie verwendet. Nach [TZITZIKAS ET AL. 2001] besteht eine Datenquelle aus zwei grundlegenden Komponenten. Die Ontologie beschreibt die Struktur und die Abhängigkeiten einer Menge von Begriffen (Termen), unter denen die Daten der Datenquelle zu finden sind. Die Daten selbst werden in einer Datenbasis (Datenbank) gespeichert. In Abbildung 2-7 ist die grundlegende Architektur des Mediatormodells und der Aufbau der Datenquellen zu sehen.

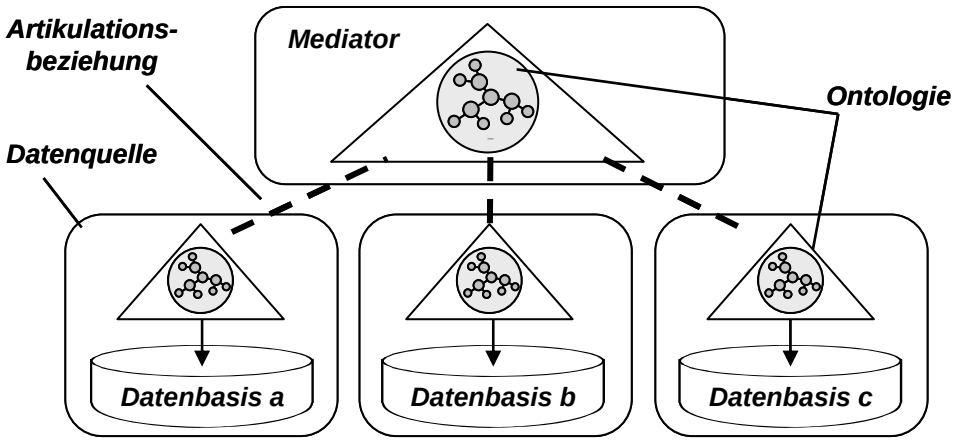


Abbildung 2-7: Mediatormodell nach [TZITZIKAS ET AL. 2001]

Die Ontologien der einzelnen Datenquellen sind unterschiedlich, wobei die einzelnen Begriffe Unterschiede in der Namensgebung sowie Granularität aufweisen können.

Für die Integration der einzelnen Datenquellen und die Definition einer einheitlichen Schnittstelle besitzt der Mediator ebenfalls eine Ontologie. Diese ist individuell auf die Kriterien und Bedürfnissen des Anwenders zugeschnitten und beschreibt meist ein konkretes Anwendungsgebiet, beispielsweise die Begrifflichkeiten für den Zahlungsverkehr im Bankwesen. Darüber hinaus besitzt der Mediator eine Menge so genannter Artikulationsbeziehungen, welche die Begriffe der verschiedenen Ontologien der Datenquellen mit jener des Mediators aufeinander abbilden. Die Begriffsdefinitionen der Mediator-Ontologie weisen die erforderliche Abstraktionsebene und Granularität auf, um alle Datenquellen für die Suche miteinzubeziehen sowie die Informationen in der gewünschten Form zurückzuliefern. Mediatoren sorgen somit dafür, dass sich die vielen einzelnen Datenquellen dem Anwender gegenüber als einzige große virtuelle Datenbasis darstellen lassen.

2.2.4 Föderierte Datenbank- und Data-Warehouse-Systeme

Ebenso wie Mediatoren fassen föderierte Datenbanken verschiedene Datenquellen zu einer einzigen, gemeinsamen Datenbasis zusammen. Im Gegensatz zum Mediator-Konzept handelt es sich bei den Datenquellen um einzelne Datenbanken. Die Bezeichnung *Föderation* unterstreicht hierbei, dass es sich um einen Zusammenschluss mehrerer, autonomer Datenbanksysteme handelt. In Abbildung 2-8 ist der prinzipielle Aufbau eines föderierten Datenbanksystems dargestellt.

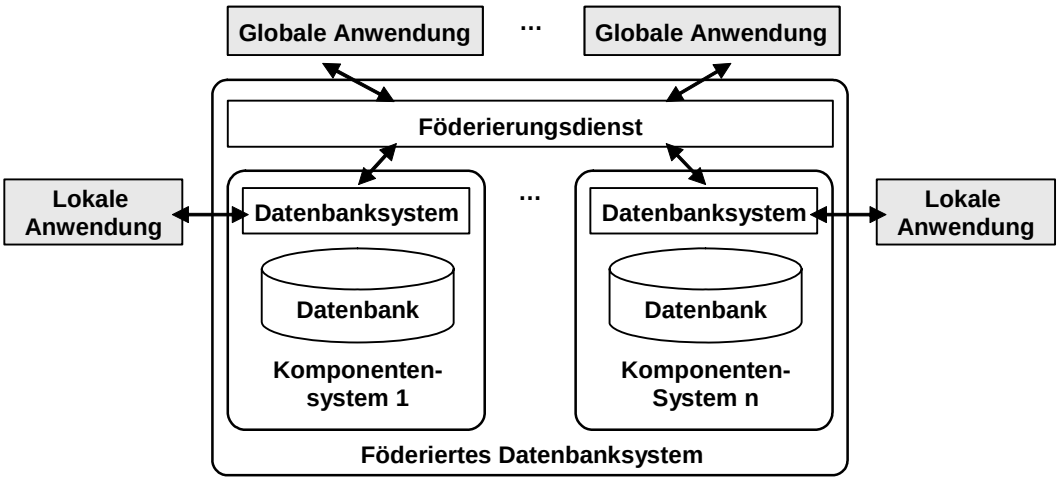


Abbildung 2-8: Systemarchitektur föderierter Datenbanksysteme [CONRAD 1997]

Die bestehenden Datenbanksysteme gehen als so genannte Komponentensysteme in das föderierte System ein. Die eigentliche Integration übernimmt ein *Föderierungsdienst*. Dieser gliedert die vom Anwender initiierten Datenabfragen (so genannte *Queries*) in einzelne Subqueries und leitet diese an die einzelnen Datenbank- bzw. Komponentensysteme weiter [GARCIA-MOLINA ET AL. 2000]. Globale Anwendungen können somit auf den Datenbestand des föderierten Systems zugreifen, als ob es eine einzige große Datenbank wäre. Da die Datenbanksysteme selbst nicht verändert oder angepasst werden müssen, können lokale Anwendungen

weiterhin auf den von ihnen benötigten Datenbestand zugreifen [CONRAD 1997]. Für die Anbindung des Föderierungssystems an die Komponentensysteme werden besondere Schnittstellen benötigt, so genannte *Wrapper*. Diese können bei Bedarf die von Komponentensystemen gelieferten Daten in ihrer Abstraktion bzw. Granularität anpassen. Die Wrapper werden häufig nach dem Mediatorenkonzept ausgeführt.

Beim Einsatz von Mediatoren sowie von föderierten Systemen werden die Daten *logisch* integriert. Das heißt, die Daten werden nicht durch die Integrationslösung gesondert gespeichert, sondern lediglich durch eine einheitliche Abfrageschnittstelle zur Verfügung gestellt. *Data-Warehouse-Systeme* hingegen führen eine *physische* Integration durch. Mit Data-Warehousing wird ein Ansatz beschrieben, eine integrierte, dauerhafte und zeitabhängige Datensammlung für unterschiedliche Analysen bereitzustellen [INMON 1992]. In Data-Warehouse-Systemen werden die benötigten Daten aus den einzelnen Datenquellen extrahiert und in einem globalen Datenmodell miteinander kombiniert. Dieses Datenmodell bildet die Grundlage für eine zentrale Datenbank, in welcher die extrahierten Daten abgespeichert werden. Diese Datenbank wird als *Data-Warehouse* bezeichnet. Das Data-Warehouse muss in regelmäßigen Abständen aktualisiert werden, um die Analyse auf gültigen Daten durchführen zu können. Für die Anbindung der Datenquellen zur Datenextraktion können Integrationskonzepte von Datenbank-Föderationen verwendet werden.

2.3 Metadaten und Metamodelle

Die im vorangegangenen Abschnitt aufgeführten Konzepte und Techniken zeigen, dass die originären Daten um zusätzliche Semantik angereichert werden müssen, um konsistent in einem gemeinsamen Kontext integriert werden zu können. Ontologien helfen dabei, Begriffsdefinitionen einzuführen, deren Bedeutung für eine bestimmte Domäne klar umrissen und somit eindeutig ist. Semistrukturierte Datenmodelle ermöglichen es, aus einer Menge heterogener Datenbestände gemeinsame Strukturen abzuleiten und zusammenzufassen. Die XML als wohl bekanntester Vertreter dieser Art erlaubt es, Texte bzw. allgemeine Daten mit zusätzlichen Annotationen auszuzeichnen, um die Interpretation der Daten zu konkretisieren. Die Annotationen können dabei aus einem ontologischen Begriffsmodell entstammen. Integrationslösungen, wie beispielsweise Mediatoren, nutzen diese Möglichkeiten, um verschiedene Datenquellen für den Anwender unsichtbar miteinander zu verbinden. Erst durch diese zusätzlich eingebrachten Daten, welche die eigentlichen, informationstragenden Daten beschreiben, ist eine maschinelle Weiterverarbeitung und Interpretation derselben möglich. In diesem Zusammenhang spricht man auch von *Metadaten*.

Metadaten

Metadaten sind Daten, die ausgewählte Aspekte anderer Daten beschreiben [STAAB 2002]. Tim Berners-Lee beschreibt sie als „maschinenlesbare Informationen über elektronische Ressourcen oder andere Dinge“ [BERNERS-LEE 1997]. Die Metadaten eines Buches oder allgemein eines Dokumentes können beispielsweise der Name des Autors, das Erscheinungsjahr sowie Angaben über den Verlag oder den Erscheinungsort sein. Für die Beschreibung von Dokumenten aller Art wurde von der *Dublin Core Metadata Initiative* (DCMI) [DCMI 2006] eine Grundmenge an Metadaten definiert und im *Dublin Core Metadata Element Set, Version 1.1* als Standard ISO 15836 zusammengefasst veröffentlicht [DCMI 2004]. Die Norm umfasst

insgesamt 15 Kernfelder, um Dokumente näher zu beschreiben. Durch diese *core elements* genannten Metadaten kann beispielsweise angegeben werden, ob es sich bei dem Dokument um Tonmaterial, Bildmaterial oder einen textbasierten Inhalt handelt. Eine detaillierte Erläuterung der 15 Kernfelder ist in [DCMI 2004] enthalten. Das *Core Metadata Element Set* wird auch als *Metamodell* bezeichnet, da es ein Modell von Metadaten darstellt, die für die Beschreibung von Dokumentattributen verwendet werden.

Nach [JECKLE 2000] wird ein Metamodell „als ein Modell, das ein Modell beschreibt", definiert. In diesem Sinne formuliert ein Metamodell Modellelemente, aus denen sich wiederum Modellbeschreibungen erstellen lassen. Mit Metamodellen lassen sich demnach Modell- bzw. Sprachkonstrukte definieren, die für die Modellierung spezifischer Anwendungsbereiche eingesetzt werden können. Am Beispiel des *Dublin Core Metamodells* lässt sich dieser Sachverhalt anschaulich verdeutlichen. Dort sind die core elements definiert, also die Sprachkonstrukte, mit denen verschiedene Dokumentattribute beschrieben werden können. Werden diese Attribute mit Werten belegt, also einem Namen für den Autor und einer Zahl für das Erscheinungsjahr, so erhält man ein *Dublin Core Modell* eines Dokumentes. In diesem Zusammenhang spricht man auch von der Instanz eines Metamodells.

Ein Metamodell beschreibt in einem weiteren Sinn die Semantik von Sprachkonstrukten sowie deren Beziehungen und gibt dadurch vor, welche Elemente auf Instanzebene zur Modellierung eines Sachverhalts verwendet werden können. Die Sprachelemente der UML (siehe Anhang 11.2.2) sind ebenfalls in einem Metamodell definiert. In den folgenden Abschnitten wird das Metamodell-Konzept der UML erläutert.

Die Metamodellarchitektur der OMG

Unabhängig von speziellen Anwendungsbereichen ist es das Ziel von Modellen, Objekte der realen Welt idealisiert abzubilden. Welche Möglichkeiten das Modell hierzu bietet, lässt sich in einem Metamodell anhand der dort spezifizierten Sprachkonstrukte festlegen. Mit Metamodellen lassen sich demnach Sprachen definieren, die für den Einsatz in speziellen Anwendungsgebieten zugeschnitten sind. So ist die UML eine Modellierungssprache zur Abbildung komplexer Softwaresysteme und besitzt die hierfür erforderlichen Sprachkonstrukte. Gerade im Hinblick auf das Thema Datenintegration ist es mitunter erforderlich, Modellinstanzen verschiedener Modellierungssprachen ineinander überführen zu können oder wenigstens Teile davon in die jeweilig andere Sprache zu übersetzen. Damit dies funktionieren kann, müssen die den Modellierungssprachen zu Grunde liegenden Metamodelle kompatibel zueinander sein. Die OMG hat hierfür eine vierstufige *Metamodell-Architektur* definiert, die in Abbildung 2-9 dargestellt ist.

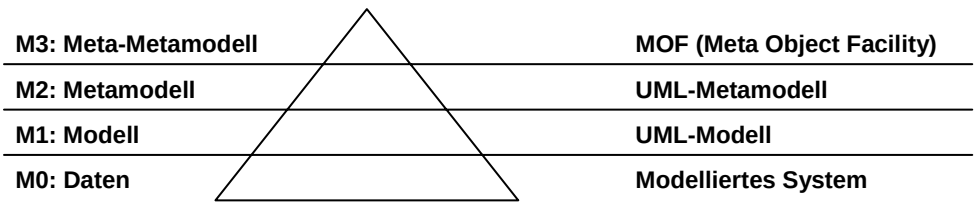


Abbildung 2-9: Die vier Modellierungsebenen der Metamodell-Architektur [OMG 2006A]

Jede Ebene enthält die Sprachkonstrukte der darunter liegenden Ebene. Die Modellierungssprache UML wird in die Ebene M1 eingeordnet. Die mit der UML erzeugten (instanziierten) Modelle befinden sich auf der Ebene M0. Das Metamodell der UML, also die Sprachdefinition, liegt in der Ebene M2. Die Metamodell-Architektur nach Abbildung 2-9 sieht vor, dass auf der Ebene M2 weitere Metamodelle definiert werden können. Neben der UML sind somit weitere Modellierungssprachen für verschiedene Anwendungsbereiche möglich. Alle Metamodelle der OMG-Metamodell-Architektur basieren wiederum auf einer einheitlichen Sprache, einem so genannten Meta-Metamodell. Die OMG stellt hierzu die *Meta Object Facility* (MOF) [OMG 2006A] zur Verfügung. MOF liegt in der Ebene M3 und definiert Sprachkonstrukte zur Abbildung von Metamodellen. Da alle MOF-Metamodelle auf denselben Sprachkonstrukten aufbauen, sind sie miteinander kompatibel. Die Definition von Transformationsregeln wird somit wesentlich vereinfacht. Für den Austausch und den Export von MOF-basierten Modellierungssprachen wird das *XML Metadata Interchange* (XMI) verwendet [OMG 2005]. Es erlaubt somit nicht nur den Austausch von UML-Modellen, sondern eignet sich für beliebige Metadaten, sofern diese mit den Mitteln des MOF abgebildet werden können.

Die in diesem Abschnitt vorgestellten Technologien werden vorwiegend im Softwarebereich eingesetzt. Für die Definition von Produktstrukturen in der mechanischen Entwicklung und Konstruktion haben sich andere Konzepte etabliert. In den folgenden Abschnitten werden entsprechende Standards und Modellierungssprachen zur Abbildung von Produktdaten vorgestellt und bestehende Ansätze aufgeführt, die einen Brückenschlag zu MOF-basierten Sprachen versuchen.

2.4 Integriertes Produktdatenmodell nach ISO 10303 (STEP)

Der bekannteste Ansatz zur Abbildung integrierter Produktdatenmodelle ist die Normenreihe ISO 10303 mit dem Titel *Industrial Automation Systems and Integration – Product Data Representation and Exchange*, besser bekannt unter dem Akronym STEP (*Standard for the Exchange of Product Data*). Die wesentliche Zielsetzung liegt in der Integration von Produktdaten in unterschiedliche rechnerunterstützte Systeme über alle Phasen des Produktlebenszyklus hinweg. STEP behandelt die Bereiche

- Modelle zur Beschreibung von Produktdaten (*Integrated Resources*, *Anwendungsprotokolle*),
- Beschreibungsmethoden (*Description Methods*),
- Implementierungsmethoden (*Implementation Methods*) sowie
- Methoden zum Konformitätstest (Conformance Testing Methodology and Framework).

STEP kann als Baukasten verstanden werden, der Grundbausteine, so genannte *Integrated Resources*, zum Aufbau von anwendungsspezifischen Produktdatenmodellen nach definierten Regeln und genormten Methoden bereitstellt.

Zur besseren Strukturierung der umfangreichen Konzepte wurde STEP in verschiedene Serien, so genannte *Parts*, gegliedert. Jeder Part wird in einem ISO-Dokument beschrieben, welches mit einem aktuellen Bearbeitungszustand gekennzeichnet wird. In Tabelle 2-1 sind einige Dokumente aufgelistet.

Klasse	Serie	Kurzbeschreibung
Umfang und Architektur	1	Überblick und fundamentale Prinzipien
Beschreibungsmethoden	11	Definition der Sprache EXPRESS
Implementierungsmethoden (21-30)	21	Format von Austauschdateien (STEP Physical File)
	22	Definition der Schnittstelle SDAI
	23	SDAI Language Binding für C++
	26	SDAI Language Binding für IDL
	27	SDAI Language Binding für Java
Testmethodik (31-40)	31	Generelle Konzepte zur Konformitätsprüfung
Integrated Resources (41-50)	41	Generelle Informationen zur Produktbeschreibung
	42	Darstellung der Geometrie
Application Protocols (201-299)	203	Darstellung konfigurierbarer Geometrie
	212	Elektrotechnische Informationen
	214	Produktlebenszyklus in der Automobilindustrie

Tabelle 2-1: Aufbau von ISO 10303 (STEP) [SELLENTIN 1999]

Die Serien 1 bis 10 behandeln allgemeine Grundlagen und den Aufbau von STEP. Insbesondere wird hier die Sprache EXPRESS (siehe Anhang 11.3.2) beschrieben, mit der die wesentlichen Konzepte der Norm modelliert werden. In den Serien 11 bis 20 werden Beschreibungs- und Spezifikationsmethoden behandelt, die Serien 21 bis 30 beinhalten Implementierungsmethoden wie beispielsweise das *STEP Physical File* (siehe Anhang 11.3.3) zum Datenaustausch mittels ASCII-Dateien oder die Schnittstelle SDAI mit den Implementierungen in verschiedenen Programmiersprachen. Die Serien 31 bis 40 erläutern Methoden und Kriterien für Konformitätstests. In den Serien 41 bis 50 werden so genannte *Integrated Resources* vorgestellt. Dies sind grundlegende Datenstrukturen, die in den *Application Protocols* der Serie 201 bis 300 zur Definition normierter Schemata für spezielle Anwendungsbereiche eingesetzt werden. Die Integrated Resources bilden den eigentlichen Kern von STEP und bestehen aus den branchen-unabhängigen Grundmodellen (*Generic Resources*) sowie den Anwendungsmodellen (*Application Resources*).

Ein Anwendungsprotokoll beschreibt die Produktdaten unter einem spezifischen Anwendungsaspekt. Deshalb muss dessen Geltungsbereich präzise definiert werden. Die ISO 10303 stellt hierfür ein Vorgehen zur Verfügung, das in Abbildung 2-10 dargestellt ist und sich in drei Punkte gliedert. In einem ersten Schritt werden die Prozesse des Anwendungsbereichs analysiert und in dem *Application Activity Model* (AAM) beschrieben. Das Aktivitätenmodell wird mit der Sprache SADT [MARCA & MCGOWAN 1987] dargestellt. Hierbei werden die Prozesse in diskrete und für den Anwendungsbereich geeignete Aktivitäten unterteilt. Für jede Aktivität werden die Größen *eingehende Datenflüsse, ausgehende Datenflüsse, Steuergrößen* und *Hilfsmittel* bestimmt. Die einzelnen Aktivitäten sind durch Informations- und Materialflüsse verbunden. Mit den Hilfsmitteln werden normalerweise Arbeitsmittel wie Werkzeugmaschinen oder Rechnersysteme bezeichnet.

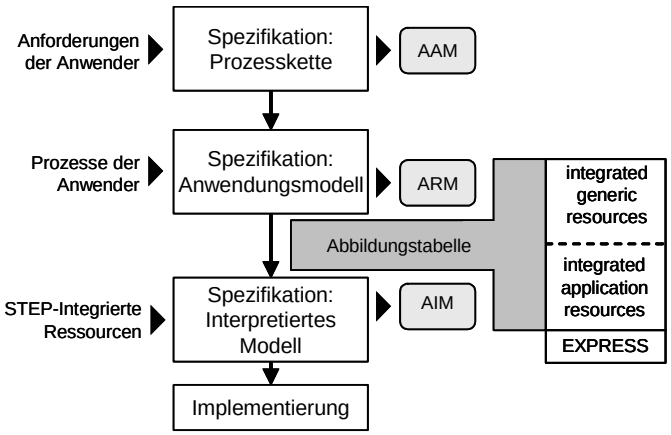


Abbildung 2-10: Bestandteile eines Application Protocols [KRASTEL 2002]

Im zweiten Schritt wird die Prozessbeschreibung im *Application Reference Model* (ARM) modelliert. Das Anwendungsdatenmodell beschreibt die Anforderungen der Anwender bezüglich des zu entwickelnden Produktmodells. Dazu werden aus Anwendersicht die im AAM identifizierten Datenklassen und ihre Beziehungen zueinander beschrieben. Hierfür wird zunehmend *EXPRESS-G* (siehe Anhang 11.3.2), eine graphisch orientierte Untermenge der Sprache EXPRESS, verwendet. Für die Implementierung wird das ARM in das *Application Interpreted Model* (AIM) überführt. Das AIM verwendet hierzu die vordefinierten Bausteine aus den Integrated Resources im Kontext des Anwendungsbereichs, für den es entwickelt werden soll. Für die Abbildung der Datenobjekte zwischen dem ARM und dem AIM wird eine Abbildungstabelle spezifiziert.

2.5 STEP, XML und UML

Der Austausch von STEP-Daten erfolgt üblicherweise mit der in der Serie 21 definierten Syntax. Dieser ASCII-basierte Ansatz findet vor allem in den konstruktionsnahen Bereichen seine Anwendung. Allerdings weist er auch einige Schwächen auf. Zum einen sind die Produktdaten im Part-21-Format nur mit großem Aufwand vom Menschen versteh- und interpretierbar und ohne spezifische Softwarewerkzeuge schwierig weiterzuverarbeiten. Zum anderen verwenden nur solche Softwarewerkzeuge dieses Format, die auf der STEP-Norm aufbauen, wie beispielsweise große CAD-Systeme [LUBELL ET AL. 2004]. In den Bereichen der Softwareentwicklung oder im *Computer Aided Control System Design* (CACSD) werden die Sprache EXPRESS oder die Serie ISO 10303-21 beispielsweise nicht eingesetzt.

Um die Anwendung von STEP auf andere Bereiche zu forcieren und die Verarbeitung von STEP-Dateien zu vereinfachen, wurde die Serie ISO 10303-28: *Implementation methods: XML Schema governed representation of EXPRESS schema governed data* in die Norm aufgenommen [ISO 10303-28 2003]. Sie ist auch unter dem Namen *STEP mark-up Language* (STEPml) bekannt. STEPml ist eine Bibliothek von XML-Spezifikationen, die auf den Informationsmodellen der ISO 10303 basieren. Sie kombiniert die in der ISO 10303 spezifizierte Semantik zur Beschreibung von Produktdatenmodellen mit den Möglichkeiten der XML zur Definition und Verarbeitung von semistrukturierten Datenmodellen. Der Vorteil von STEPml liegt vor allem in der großen Anzahl freiverfügbarer XML-Softwarewerkzeuge. Dadurch können STEP-Daten, die im STEPml-Format vorliegen, auch von Softwaresystemen ohne direk-

ten Bezug zur ISO 10303 verarbeitet werden. Dieser Aspekt wird beispielsweise in [PAN & SMITH 2003] genutzt, um aus einer XML-STEP-Datei geometrische Daten für die Montageplanung herauslesen zu können. Hierzu wird eine STEP-Datei mit der Serie-21-Syntax aus einem CAD-System ausgelesen, nach XML konvertiert und in einem Planungswerkzeug weiterverarbeitet.

Die zunehmende Bedeutung der XML als Beschreibungssprache für Produktdaten zeigt sich auch in den aktuellen Arbeiten zur Serie 25 *EXPRESS to OMG XMI binding* der ISO 10303. Hier wird ein Konzept erarbeitet, um eine bidirektionale Abbildung von Produktdaten in EXPRESS und UML-Entitäten zu ermöglichen. Ziel ist es, STEP-Daten im XMI-Format, dem neutralen Austauschformat von UML-Modellen, speichern zu können. Damit ist es nicht nur möglich, komplexe EXPRESS-Datenmodelle mit der UML graphisch darzustellen, sondern darüber hinaus mit weiteren Diagrammtypen aus der UML zu erweitern. Dieser Brückenschlag zwischen der Mechanik- und der Softwareentwicklung stellt gerade für den Bereich der Mechatronik einen bedeutenden Schritt dar. Der derzeitige Bearbeitungszustand der Serie wird mit *Working Draft*, also einem Entwurf, angegeben.

3 Stand der Technik in der Werkzeugmaschinenentwicklung

3.1 Übersicht

In diesem Kapitel werden Modellierungstechniken, Vorgehensweisen und Entwicklungsumgebungen vorgestellt, die den Stand der Technik im Werkzeugmaschinenbau repräsentieren. Um diese im Gesamtkontext der Werkzeugmaschinenentwicklung einordnen zu können, wird im Anschluss das *System* Werkzeugmaschine erläutert und ein Überblick über den allgemeinen Entwicklungsprozess gegeben. Das Kapitel schließt mit einer kritischen Betrachtung der aufgeführten Lösungsvorschläge und leitet den Handlungsbedarf für diese Arbeit ab.

3.2 Das technische System Werkzeugmaschine

Spanende Werkzeugmaschinen sind Arbeitsmaschinen zum Bearbeiten überwiegend metallischer Werkstoffe nach einer vorher bestimmten Geometrie unter Verwendung geeigneter Werkzeuge. Die Formgebung erfolgt dabei durch eine gesteuerte Relativbewegung zwischen Werkstück und Werkzeug [MILBERG 1992]. Die Genauigkeit und Geschwindigkeit dieser Relativbewegung bzw. Vorschubbewegung [DIN 6580 1985] hängt von zahlreichen Faktoren ab und entscheidet maßgeblich über die Qualität der Werkzeugmaschine [BAUDISCH 2003]. Die Vorschubbewegung zur Geometrieerzeugung erfolgt unter den Vorgaben eines Programms [DIN 66025 1981], das die Bearbeitungsaufgabe für ein bestimmtes Werkstück beschreibt.

Die Werkzeugmaschinensteuerung, die das Programm interpretiert, setzt sich aus den Modulen NC-Steuerung (Numerical Control) und speicherprogrammierbare Steuerung (SPS, engl. *Programmable Logic Controller*, PLC) zusammen (siehe Abbildung 3-1).

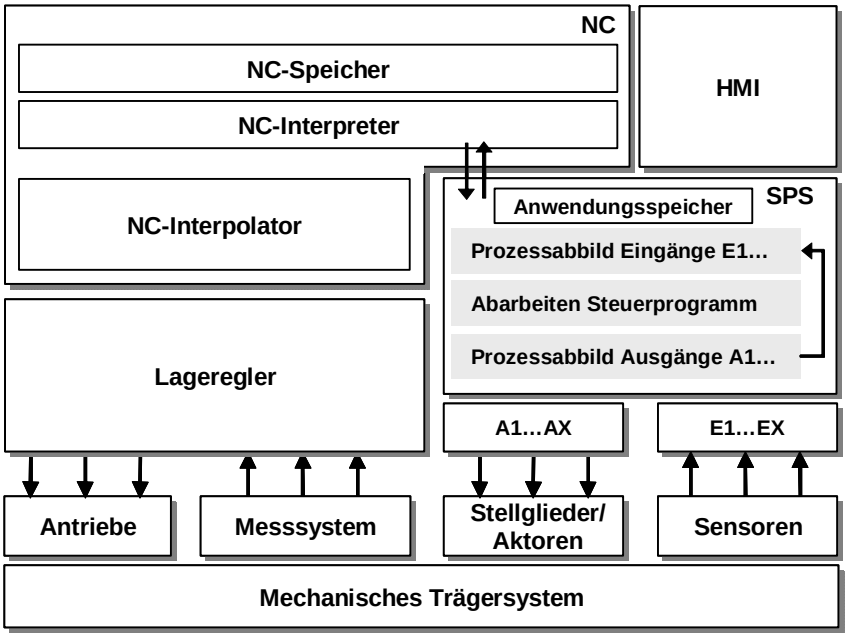


Abbildung 3-1: Systemtechnischer Aufbau einer Werkzeugmaschine

Die zentrale Aufgabe der NC-Steuerung liegt in der Umsetzung der Relativbewegung zwischen Werkzeug und Werkstück. Der NC-Interpreter liest das NC-Programm und die darin enthaltenen Unterprogramme satzweise ein und analysiert die dort enthaltenen Weg- und Geschwindigkeitsangaben sowie zusätzlich technologische Anweisungen und Schaltbefehle, wie z. B. bezüglich Schnittgeschwindigkeit, Vorschubgeschwindigkeit, Werkzeugwechsel und Kühlschmiermittelzufuhr. Geometrieanweisungen gehen an den Interpolator, während Schaltbefehle an die SPS weitergeleitet werden. Für eine gezielte Ausführung der NC-Programme ist eine Synchronisation zwischen der SPS und der NC-Steuerung erforderlich, da Achsbewegungen und Schaltvorgänge zeitlich aufeinander abgestimmt werden müssen [KREUSCH 2002].

Die SPS übernimmt bei Werkzeugmaschinen die Funktion einer Anpasssteuerung, die alle von der NC an die Maschine ausgehenden Schaltfunktionen nur dann wirksam werden lässt, wenn sie ohne Gefährdung von Mensch, Maschine und Werkstück ablaufen können [KIEF 2005]. Die SPS-Software realisiert Steuerungs-, Benutzungs- und Diagnosefunktionen, wobei über Steuerungseingänge E1,…, EX und Steuerungsausgänge A1,…, AX Schnittstellen zum Benutzer und zum Fertigungsprozess vorhanden sind [LUTZ 1999]. An den Eingängen sind üblicherweise binäre Näherungssensoren, Druckschalter und Bedientasten des *Human Machine Interface* (HMI) angeschlossen. Die Ausgänge steuern schaltende Aktoren und Stellglieder, wie Hydraulikventile oder Elektromotoren. Wegen der geringen Strombelastbarkeit der Ausgänge erfolgt dies häufig über Relais und Schütze [WAGNER 1997]. Die Anwenderprogramme der SPS werden zyklisch abgearbeitet. Nach dem Einschalten durchläuft die SPS zunächst eine Initialisierungsphase, in der entsprechend den Voreinstellungen der Anwendungsspeicher vorbelegt wird. Danach wird das Prozessabbild der Eingänge eingelesen, das Steuerungsprogramm abgearbeitet und abschließend das Prozessabbild der Ausgänge übertragen. Bei der Abfrage der Signalzustände im Steuerungsprogramm wird auf das Prozessabbild der Eingänge zugegriffen [HENGEL 1993].

Die Lageregler nehmen die vom NC-Interpolator kommenden Werte als Lagesollwerte auf. Durch den Vergleich mit den durch die Messsysteme bereitgestellten Lageistwerten wird der Schleppabstand errechnet und über den Lageregler die Stellgröße für die Drehzahl der Antriebe ermittelt. Regelungen von NC-Achsen werden digital als kaskadierte Regelkreise mit unterlagerten Drehzahl- und Stromregelkreisen sowie Sollwertfiltern und -begrenzern realisiert und setzen die errechneten Soll-Drehzahlwerte in entsprechende Stromwerte um [ZÄH ET AL. 2004]. In den Vorschubachsen werden Drehstrom-Synchronmotoren, die nach dem elektrodynamischen Prinzip die Vorschubmomente erzeugen, eingesetzt. Ihre elektrische Leistung beziehen diese aus einem Leistungsverstärker mit Gleichspannungszwischenkreis und Wechselrichter, die als Stellglieder der Lageregelkreise fungieren. Zur Aufnahme der Messgrößen von Lagen und Geschwindigkeiten werden neben induktiven Messprinzipien überwiegend optische Sensoren mit zeit- und amplitudendiskreter Signalabtastung verwendet [GROßMANN & ARNDT 2000]. Die Aktoren und Sensoren einer Werkzeugmaschine werden auch als elektrische Betriebsmittel bezeichnet. Sie bilden letztendlich die Schnittstelle zwischen der Steuerung und dem technologischen Fertigungsprozess.

Als Arbeitsmaschinen basieren Werkzeugmaschinen immer auf einem mechanischen Trägersystem, da die Abstützung bzw. Übertragung der Prozesslasten nur durch mechanische Prin-

zipien realisierbar ist. Zwischen der Erzeugung der Führungsgrößen durch den Interpolator und ihrem Bestimmungsort an der Prozessstelle liegt eine informationstechnisch verzweigte mechatronische Übertragungsstrecke mit regelungstechnischen, elektrischen und mechanischen Teilstrecken (siehe Abbildung 3-2).

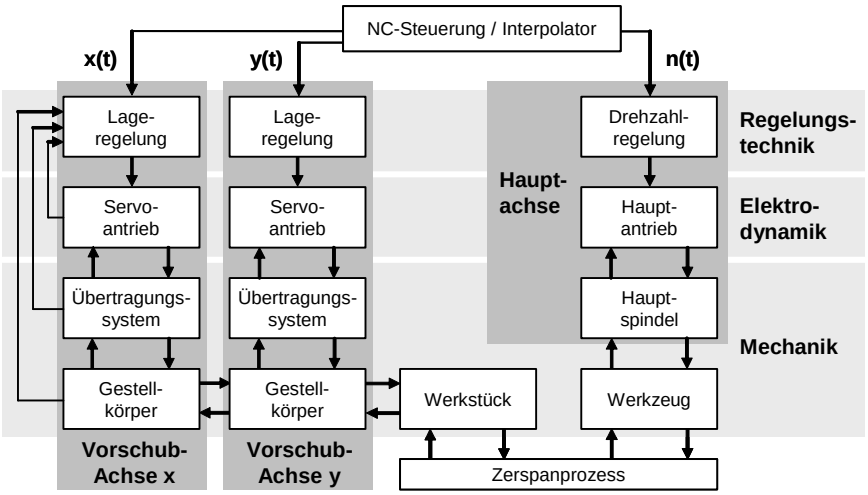


Abbildung 3-2: Technologisches Wirkprinzip von spanenden NC-Maschinen [REINHART ET AL. 2001C]

Aufgrund der Rückführung von Messgrößen an die Antriebsregelung und der Rückwirkung des Zerspanprozesses auf die mechanische Struktur liegen mehrfache Rückkopplungen innerhalb des Gesamtsystems vor. Die Prozesslasten aus den Antrieben und dem Werkzeug-Werkstück-Eingriff rufen Kraftreaktionen an den Koppelstellen der mechanischen Komponenten hervor. Die Verbindungs- und Übertragungselemente (Führungen, Lager, Spindeln) sowie die Gestellkörper selbst erfahren Deformationen statischer und dynamischer Art. Die damit verbundenen Schwingungen des mechanischen Systems überlagern sich einerseits mit den Messwerten der Sensorsysteme und begrenzen die Regelkreisdynamik [HAMAN & TRÖNDLE 1997], andererseits besteht die Gefahr der regenerativen Rückkopplung mit den Zerspankräften [WECK & TEIPEL 1977]. Diese Wechselwirkungen müssen zur Erreichung hoher Leistungs- und Genauigkeitsanforderungen möglichst frühzeitig erkannt und die Teilsysteme optimal aufeinander abgestimmt werden [ZÄH ET AL. 2004].

Die systemtechnische Betrachtung moderner Werkzeugmaschinen zeigt deutlich, wie sich die einzelnen Subsysteme beeinflussen. Um eine hohe Dynamik bei einer ausreichenden Genauigkeit ermöglichen zu können, müssen der technologische Fertigungsprozess, das Regelungs- und Steuerungskonzept sowie der mechanische Grundaufbau durchgängig aufeinander abgestimmt werden. Zur Abbildung der komplexen Zusammenhänge zwischen diesen Bereichen werden in den einzelnen Arbeitsschritten unterschiedliche Modellierungstechniken verwendet. Der folgende Abschnitt gibt einen Überblick über die Werkzeugmaschinenentwicklung und ermöglicht die Zuordnung der eingesetzten Modellbeschreibungen zu den einzelnen Aufgabenbereichen.

3.3 Vorgehensweise bei der Entwicklung von Werkzeugmaschinen

Der in dieser Arbeit fokussierte Ausschnitt des Entwicklungsprozesses für spanende Werkzeugmaschinen ist in Abbildung 3-3 dargestellt. Das Bild zeigt die Skizze einer allgemeingültigen Vorgehensweise und stellt die wesentlichen Aufgabenbereiche dar.

Der Ausgangspunkt für die Entwicklung der Maschine ist die Auftragsbeschreibung. Der Kunde als zukünftiger Betreiber erstellt hierfür das Lastenheft, das zusammen mit dem Pflichtenheft sowohl als fachliche als auch als rechtliche Grundlage für alle Projektbeteiligten gilt. Darüber hinaus können aus dem Vertrieb bzw. aus spezifischen Marktanalysen Anforderungen an die Maschineneigenschaften gestellt werden und in die Entwicklung miteinfließen.

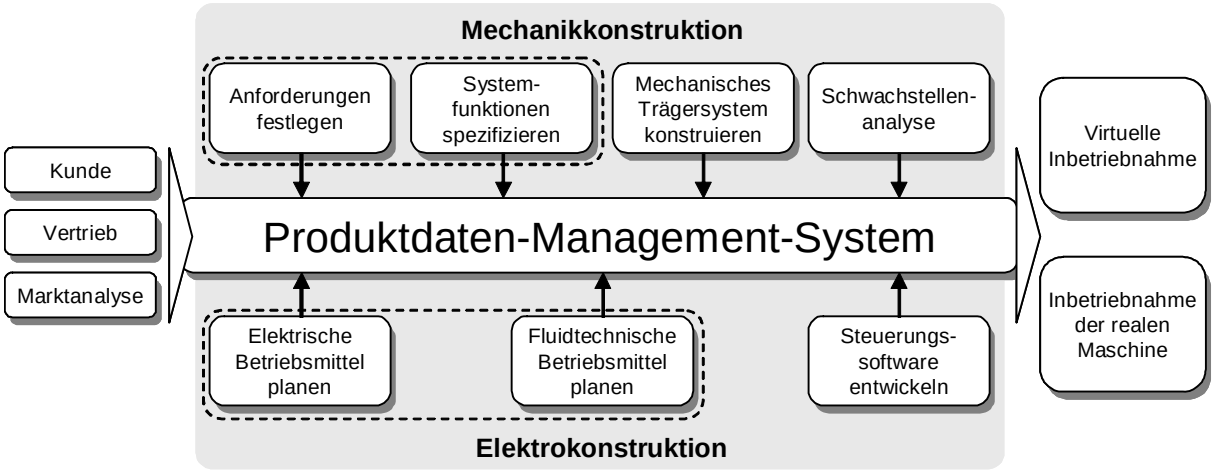


Abbildung 3-3: Entwicklungsprozess im Werkzeugmaschinenbau

Die Mechanikkonstruktion erarbeitet aus der Auftragsbeschreibung bzw. dem Lastenheft die Anforderungen an die Maschine. Diese sind zum Teil sehr detailliert und können spezifische Vorgaben über den Hersteller einzelner Bauteile oder die Lackierung bzw. Beschriftung von Plaketten nach firmeninternen Standards enthalten. Aus diesen Informationen ermittelt die Mechanikkonstruktion die Funktionseinheiten, die Maschinenkomponenten und die Maschinenstruktur (siehe Abschnitt 3.4.1). Mithilfe geeigneter CAD-Softwarewerkzeuge werden Konstruktionszeichnungen erstellt und zusammen mit der Elektrokonstruktion werden die elektrischen, hydraulischen und pneumatischen Betriebsmittel geplant und in das mechanische Trägersystem (Maschinenbett, Maschinenständer) integriert (siehe Abschnitt 3.5.1). Bei der Schwachstellenanalyse werden strukturdynamische Eigenschaften wie das Schwingungsverhalten untersucht und gegebenenfalls durch geeignete konstruktionstechnische oder regelungstechnische Modifikationen optimiert (siehe Abschnitt 3.5.2 und Abschnitt 3.4.4). Bei der Steuerungsprojektierung (siehe Abschnitt 3.4.3) wird die Software zur Steuerung der Maschine entwickelt. Grundlage hierfür sind unter anderem die zuvor erstellten Dokumente zur Funktionsbeschreibung und zur elektrischen bzw. fluidtechnischen Betriebsmittelplanung. Der Test und die Korrektur der Software erfolgt an der Maschine selbst bzw. bei der Inbetriebnahme vor Ort, sofern keine entsprechenden Simulationssysteme für die so genannte „virtuelle Inbetriebnahme“ vorhanden sind. Hierbei werden alle notwendigen Maschinenbaugruppen und deren physikalisches Verhalten am Rechner abgebildet und die Steuerungssoft-

ware mit Hilfe dieses Modells getestet (siehe Abschnitt 3.5.3). Die bei der Entwicklung der Werkzeugmaschine erstellten Entwicklungsdokumente, Pläne und Zeichnungen werden vorwiegend in einem zentralen System zur Verwaltung von Produktdaten gespeichert und administriert, dem Produktdaten-Management-System (siehe Abschnitt 3.6).

3.4 Modellkonzepte und Vorgehensweisen

Der Anteil elektronischer und informationstechnischer Komponenten in der Werkzeugmaschine nimmt stetig zu. Integriert in rein mechanischen Funktionsstrukturen entstehen komplexe mechatronische Teilsysteme (siehe Abbildung 3-4). Die komplexen Zusammenhänge zwischen diesen Teilsystemen müssen unter anderem bei der Entwicklung der Maschinendynamik und der Automatisierungskomponenten beherrscht werden. So sind beispielsweise die starken Wechselwirkungen zwischen der Gestellstruktur und der Antriebs-, Steuerungs- und Messtechnik ohne rechnertechnische Unterstützung nicht mehr gezielt zu beeinflussen. Ebenso sind bei der Automatisierung von Werkzeugmaschinen innovative Softwarewerkzeuge und Modellkonzepte zur Beschreibung, Implementierung und Umsetzung von Maschinenabläufen erforderlich.

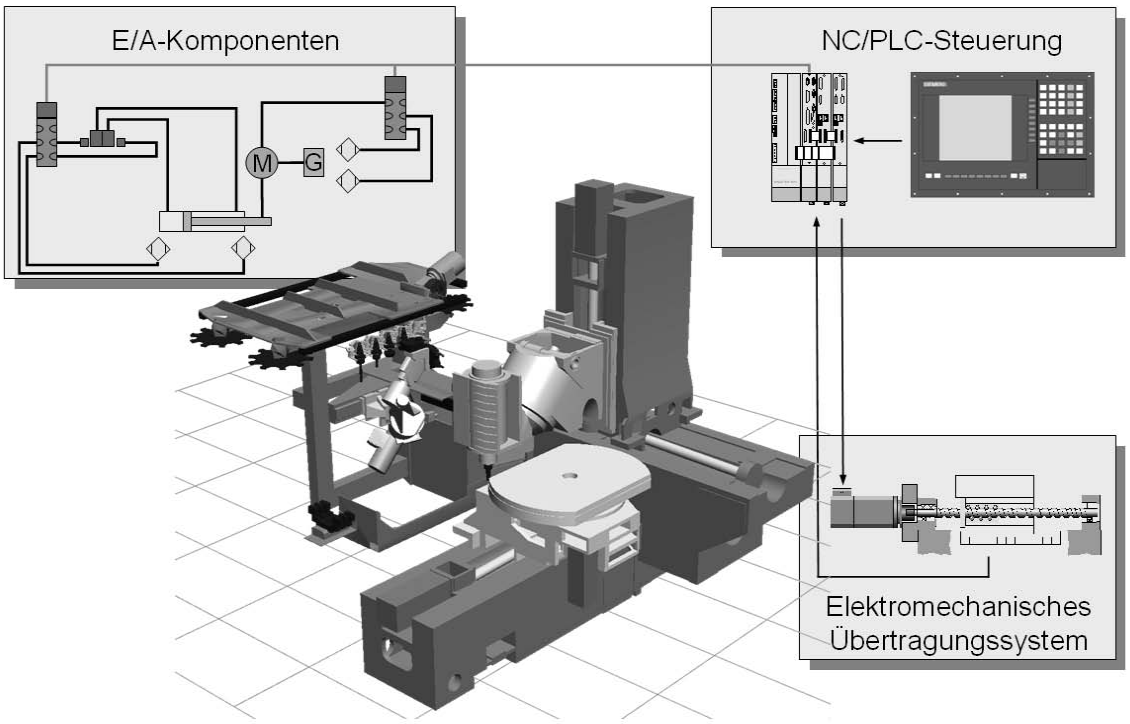


Abbildung 3-4: Das mechatronische System Werkzeugmaschine

Der zunehmende Einsatz von Modellbeschreibungen wird unter dem Begriff der *Modellgestützten Entwicklung* bzw. des *Model Driven Engineering* zusammengefasst. Die Grundlage hierfür stellen vom Rechner algorithmisch erfass- und interpretierbare Modellierungstechniken dar. Auf diese Weise wird der Konstrukteur durch automatisierte Konsistenzchecks und Generatoren zur Modellerstellung und Projektdokumentation in seiner Entwicklungstätigkeit

unterstützt. Dieses Vorhaben kann jedoch nur dann effizient umgesetzt werden, wenn geeignete Techniken vorhanden sind, welche die eingesetzten Softwarewerkzeuge bzw. die dahinter stehenden Modellkonzepte in einem gemeinsamen Rahmenkonzept zusammenführen. Insbesondere die Integration der einzelnen Modellkonzepte und deren Umsetzung in Softwaresystemen gewinnt zunehmend an Bedeutung, um das gerade in der Mechatronik so wichtige gesamtheitliche Systemverständnis fördern zu können. Das Rahmenkonzept muss dabei den Entwicklungsprozess (Vorgehensmodell), die eingesetzten Modellierungstechniken und Beschreibungskonzepte sowie die verwendeten Softwarewerkzeuge umfassen. In den folgenden Abschnitten werden die aus Forschung und Industrie bekannten Ansätze aufgeführt und in Hinblick auf die Zielsetzung dieser Arbeit diskutiert und bewertet.

3.4.1 Beschreibung von Systemfunktionen

Die Standardwerke von [PAHL ET AL. 2004], [EHRLENSPIEL 1995], [ROTH 2001] und [KOLLER 1985] zum Thema Funktionsbeschreibung und Konstruktionsmethodik dokumentieren allgemein gültiges Basiswissen und werden für diese Arbeit als gegeben vorausgesetzt. Die dort postulierten Aussagen über grundlegende Vorgehensweisen bei der Lösungsfindung, Funktionsbeschreibung und dem Einsatz von Konstruktionskatalogen werden durch die in diesem Abschnitt vorgestellten Arbeiten aufgegriffen und zum Teil für spezifische Anwendungsbereiche konkretisiert. Insbesondere verweisen diese darauf, dass die Grundlage für die arbeitsteilige Bearbeitung von Konstruktions- und Entwicklungstätigkeiten die präzise Formulierung der Aufgabenstellung darstellt. Aus den dort formulierten Anforderungen werden im Lösungsprozess schrittweise konkrete Systemmerkmale abgeleitet und in der Beschreibung der einzelnen Systemfunktionen konkretisiert. Nach [BIRLI ET AL. 1997] reicht für komplexe mechatronische Systeme eine getrennte Analyse und Synthese der einzelnen Subsysteme nicht aus. Im Zentrum einer funktionalen Betrachtung steht deshalb die Gliederung der Gesamtfunktion in Teilfunktionen und deren Zuordnung zu Teilstrukturen.

Anforderungen abbilden

Die Anforderungen werden vom Auftraggeber im Lastenheft zusammengefasst. Der Auftragnehmer nimmt dieses Dokument als Grundlage und fasst seinerseits Anforderungen für die technische Umsetzung im Pflichtenheft zusammen. Dieses enthält produktbezogene Informationen wie beispielsweise Daten über den Bauraum, Material oder Anschlussmaße sowie organisatorische bzw. wirtschaftliche Randbedingungen. Wie die VDI-Richtlinie [VDI 2221 1993] verdeutlicht, stellt das Pflichtenheft einen phasen- und fachbereichsübergreifenden Wissensspeicher dar, welcher im Entwicklungsprozess laufend erweitert und verändert wird. Deshalb ist es von großer Bedeutung, Anforderungen transparent und widerspruchsfrei zu dokumentieren sowie zielgerichtet an die einzelnen Fachbereiche weiterzuleiten.

In den Arbeiten von [AHRENS 2000] und [UHLIG 2002] werden neue Ansätze vorgestellt, um Anforderungen aufnehmen und verwalten zu können. Der Schwerpunkt beider Arbeiten liegt in der direkten Übernahme der Anforderungen in spätere Arbeitsschritte in Form von automatisch generierten Lasten- und Pflichtenheften. Neben den methodischen Aspekten werden auch mögliche Softwarewerkzeuge für das Anforderungsmanagement erarbeitet. In [AHRENS 2000] wird ausgeführt, dass eine effektive Unterstützung des Arbeitens mit Anforderungen

nur durch ein Zusammenwirken von Methodik und Informationstechnologie erfolgen kann. Für die IT-unterstützte Transformation von Kundenanforderungen in technische Anforderungen werden bekannte Modelle wie *Netzpläne*, *Balkenpläne*, *QFD-Matrizen* oder *Conjoint-Analysen* eingesetzt. Die Umsetzung der erarbeiteten Konzepte erfolgt im Softwarewerkzeug *SPECTool*, welches innerhalb einer Lotus-Notes-Umgebung läuft [IBM 2005]. Über spezielle Eingabemasken können gemäß der erarbeiteten Anforderungsstrukturen und Klassifizierungen Anforderungen eingegeben und als MS-Office-Dokumente exportiert werden. Durch die Definition eines Regelwerkes werden unvollständige Anforderungsspezifikationen erkannt.

Mit einer ähnlichen Thematik befasst sich auch [UHLIG 2002]. Hier wird jedoch die Notwendigkeit formalisierter Modelle zur rechnergestützten Transformation von Anforderungen in zugehörige Lösungselemente stärker betont. Zielsetzung seiner Arbeit ist es, Methoden und Werkzeuge bereitzustellen, um die Zusammenarbeit zwischen Auftraggeber und Auftragnehmer bei einer unternehmensübergreifenden Produktentwicklung zu verbessern. Der Fokus liegt dabei auf dem Bereich der Mechanikkonstruktion. Die in den frühen Phasen der Entwicklung definierten Anforderungen sollen dem Konstrukteur bei der Detaillierung im CAD-Werkzeug zur Verfügung stehen. Randbedingungen wie zu verwendende Werkstoffe oder geometrische Vorgaben sollen über eine Schnittstelle zur Verfügung gestellt werden. Um dies zu ermöglichen, wird eine Kommunikationsplattform zum redundanzfreien Austausch von Lastenheft, Pflichtenheft, Angebot, CAD-Daten und den jeweiligen Änderungen erarbeitet und vorgestellt. Die Plattform ermöglicht eine variable Beschreibung von Entwicklungsvorgaben, welche durch ein Änderungsmanagement den sich stetig ändernden Randbedingungen angepasst werden können. Die von der Kommunikationsplattform verwalteten Dokumente basieren auf einem zentralen Datenmodell und werden als XML-Dokumente auf einem Server gespeichert. Die Arbeit mit den Dokumenten erfolgt anhand eines Modellservers, der wiederum über einen Verwaltungsserver auf die zentral gespeicherten Dokumente und Modelle zugreift. Der Verwaltungsserver arbeitet mit einer Verzeichnisstruktur auf Dateiebene und erfasst jedes Dokument und Modell in einer entsprechenden Verwaltungsstruktur. Am Beispiel der Kopplung der entwickelten Kommunikationsplattform mit einem 3D-CAD-System wird die Tragfähigkeit des Konzeptes nachgewiesen. Die Verwaltung von Anforderungen für die Softwareerstellung oder die Elektroplanung war nicht Schwerpunkt der Arbeit und wird dementsprechend vom Softwareprototyp nicht unterstützt.

In den Arbeiten von [DEIFEL 2001] und [GEISBERGER 2005] werden Lösungsansätze vorgestellt, die auf das Requirementsengineering im IT-Bereich fokussieren. Kern der Arbeit von [DEIFEL 2001] ist ein Modell von Entwicklungsdokumenten und ein darauf abgestimmtes Prozessmodell zur systematischen Unterstützung des Requirementsengineerings komplexer Standardsoftware. Das Prozessmodell definiert ein Vorgehen für das Herausarbeiten, die Verhandlung und die Spezifikation von Anforderungen. Das Modell der Entwicklungsprodukte beschreibt zu erarbeitende Ergebnisse sowie methodische Vorlagen und Anleitungen. Zielsetzung der Arbeit von [GEISBERGER 2005] ist die Definition einer methodischen Grundlage für die interdisziplinäre Erarbeitung der Anforderungs- und Systemspezifikation eingebetteter Systeme.

Funktionsstrukturen für Systeme beschreiben

Die Umsetzung von Anforderungen in transparente interdisziplinäre Funktionsmodelle wird in [KALLMEYER 1998] behandelt. Hier wird eine Beschreibungssprache und eine Vorgehensweise für die Prinziplösungsmodellierung mechatronischer Systeme vorgestellt.

In Abbildung 3-5 ist ein Ausschnitt der graphischen Notation der entwickelten Beschreibungssprache dargestellt.

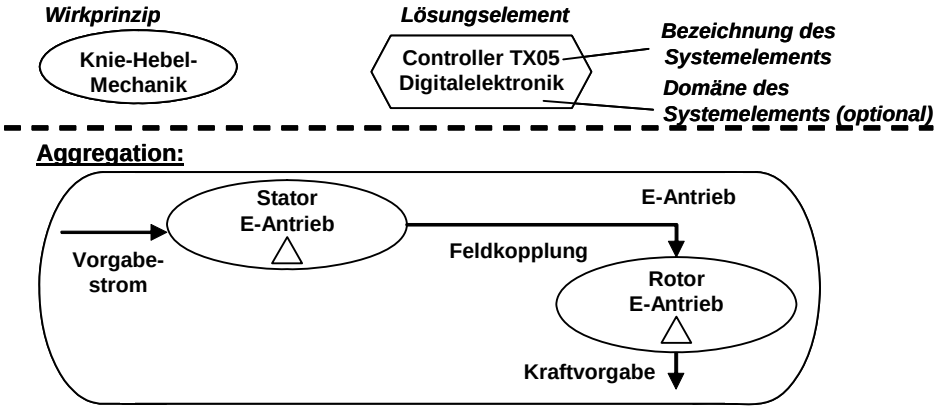


Abbildung 3-5: Sprachkonstrukte zur Prinziplösungsmodellierung mechatronischer Systeme [KALLMEYER 1998]

Ein Wirkprinzip ist ein Konzept, das eine mögliche Umsetzung einer Funktion, beispielsweise durch einen physikalischen Effekt oder einen logischen Ablauf, grob spezifiziert. Ein Lösungselement hingegen ist eine bereits bewährte konkrete Realisierung zur Erfüllung einer Funktion. Wirkprinzipien und Lösungselemente können zu einer Aggregation zusammengefasst werden, wobei logische oder räumliche Gesichtspunkte beachtet werden müssen. In [FLATH 2002] werden die Modellierungselemente aus [KALLMEYER 1998] um die Möglichkeit zur Abbildung von Störfunktionen und Störzuständen erweitert. Ausgehend von den Produktanforderungen werden mit Hilfe der Modellelemente *Zustand*, *Störzustand*, *Sollfunktion* und *Störfunktion* mechatronische Prinziplösungen systematisch aufgebaut. Eine informationstechnische Umsetzung der Modellierungssprache wird als zukünftiger Arbeitsbereich aufgeführt. Insbesondere wird in diesem Zusammenhang auf die Kopplung bestehender Softwarewerkzeuge zur Funktions- und Verhaltensmodellierung als mögliche Umsetzungsvariante hingewiesen.

In eine ähnliche Richtung gehen die Arbeiten von [LANGLOTZ 2000] und [HUANG 2002]. Das Funktionsmodell wird in beiden Arbeiten als Verbindung von Aufgabenstellung und Lösungsprinzipien definiert. Im Gegensatz zu [KALLMEYER 1998] liegt der Schwerpunkt auf einer wissensbasierten Funktionsmodellierung und Lösungsfindung mechatronischer Systeme. Für die Definition der eigenen Funktionsstrukturen verwendet [LANGLOTZ 2000] die in [ROTH 2001] erläuterten allgemeinen Basisfunktionen zur Systembeschreibung. Deren Verhalten wird anhand definierter Verben präzisiert und durch das Zusammenwirken der drei Kategorien *Stoff*, *Energie* und *Information* zu Funktionsstrukturen verknüpft. In der Arbeit von [LANGLOTZ 2000] werden diese Basisfunktionen um weitere Funktionen ergänzt und an-

hand von *Taxonomien* klassifiziert. Taxonomien stellen eine hierarchische Abbildung von Begrifflichkeiten einer bestimmten Domäne bzw. eines Fachbereichs dar. Die Klassifizierung erfolgt nach den allgemeinen Funktionsverben nach [ROTH 2001] und nach dem Änderungscharakter der Funktion, je nachdem ob es sich um eine *orts-verändernde*, *informationsverändernde* oder *zeit-verändernde* Funktion handelt. Ebenso werden die Elemente der drei Kategorien *Stoff*, *Energie* und *Information* mittels einer taxonomischen Gliederung hierarchisiert. Die Taxonomien bilden die Basis für ein Informationsmodell, das als Wissensbasis für die Funktionsmodellierung verwendet wird. Der entwickelte Funktionsmodellierer *DICAD-ASP* nutzt dieses Informationsmodell zum Aufbau und zur Verwaltung der Funktionsmodelle. Auf dieser Grundlage wird der Anwender beim Aufbau geeigneter Funktionsstrukturen unterstützt und gegebenenfalls auf Unstimmigkeiten aufmerksam gemacht. Der Ansatz zeigt die Möglichkeiten eines Informationsmodells bei der semantischen Prüfung von Funktionsmodellen auf. Das zu Grunde liegende Konzept der Wirkprinzipien nach [ROTH 2001] legt den Schwerpunkt der Arbeit in die Modellierung bzw. Realisierung physikalischer Effekte. Für ein von der Mechatronik geprägtes Entwicklungsumfeld, wie im Bereich des Werkzeugmaschinenbaus, ist dieser Ansatz deshalb nur bedingt geeignet. Die Abbildung von Funktionen für die Steuerungssoftware und deren Rückwirkung auf Signalgeber im Maschinenfeld kann nur unzureichend gewährleistet werden.

Auch in [HUANG 2002] liegt der Schwerpunkt auf einer wissensbasierten Funktionsmodellierung und Lösungsfindung mechatronischer Systeme. Insbesondere wird in der Arbeit das grundlegende Zusammenspiel der interdisziplinären Größen *Energie*, *Stoff* und *Information* untersucht und in einem Datenmodell in UML-Notation dokumentiert.

Die aufgeführten Arbeiten für die Erstellung von Funktionsstrukturen beziehen sich auf mechatronische Systeme und umfassen somit auch Werkzeugmaschinen. Die Explizierung von Funktionen in gesonderten Modellen ist in der Werkzeugmaschinenentwicklung jedoch nicht üblich, sondern erfolgt aus Zeitgründen implizit im Rahmen der Mechanikkonstruktion. Im folgenden Abschnitt werden die Nutzenpotenziale einer Funktionsbeschreibung bei der Konfiguration von Werkzeugmaschinen aus Komponentenbibliotheken erläutert.

3.4.2 Konfiguration von Werkzeugmaschinen

Die zuvor erläuterten Funktionsmodelle bilden die Grundlage für die Konfigurierung von Werkzeugmaschinen anhand von Bibliotheken. In [BÖGER 1998] wird ein Referenzmodell zur Modularisierung von Werkzeugmaschinenkonzepten vorgestellt, welches auf einer funktionalen Dekomposition basiert. Den Ausgangspunkt bilden so genannte *Basisfunktionen*, welche eine Werkzeugmaschine zur Ausführung eines Fertigungsprozesses umsetzen muss, also beispielsweise *Werkzeugführung*, *Werkstückführung* oder *Werkstückbereitstellung*. Die Basisfunktionen lassen sich in Bezug auf ihre Integration in den Kraftfluss des Bearbeitungsprozesses in *Haupt-* und *Nebenfunktionen* untergliedern. Hauptfunktionen liegen im Kraftfluss, der durch den Zerspanprozess induziert und durch die Maschine geleitet wird, wohingegen Nebenfunktionen außerhalb desselben liegen. Alle Module bzw. Baugruppen einer Werkzeugmaschine können einer oder mehreren dieser Basisfunktionen zugeordnet werden. Das Referenzmodell sieht nun vor, dass ausgehend von einer Anforderungsspezifikation die erforderlichen Basisfunktionen und in einem zweiten Schritt die entsprechenden Module bestimmt

werden. Das Referenzmodell ist in Anlehnung an die ISO 10303 (STEP) realisiert und in der Spezifikationssprache EXPRESS-G dargestellt. Für die Konfigurierung konsistenter Maschinenmodelle werden Parameter zu den Basisfunktionen definiert und im STEP-Modell in einer taxonomischen Hierarchie abgebildet. Das Referenzmodell lieferte die Grundlage für einen Softwareprototypen, mit dem die Praxistauglichkeit des Modularisierungskonzeptes an einer bestehenden Werkzeugmaschine untersucht wurde. Der Ansatz bietet einen interessanten Vorschlag zur funktionalen Dekomposition von Werkzeugmaschinen. Eine ähnliche Vorgehensweise zur einhergehenden Modularisierung der Funktionen der Steuerungssoftware wird jedoch nicht explizit aufgeführt.

Eine im Sinne des Model Driven Engineering auf Modellbeschreibungen basierende Integration von neuen bzw. erweiterten Funktionalitäten in der Werkzeugmaschinenentwicklung bildet den Schwerpunkt in [NEUHAUS 2003]. Als Voraussetzung für die Konfiguration von Werkzeugmaschinen werden im Wesentlichen Modularität und geeignete Konfigurationshilfen genannt. Als Grundlage hierfür wird eine Referenzarchitektur modularer Werkzeugmaschinen eingeführt. Dieses in EXPRESS-G abgebildete Modell wird in einem Top-Down-Ansatz weiter detailliert und dient dem Software-Prototypen des Konfigurierungsmanagers als Datenmodell (siehe Abbildung 3-6).

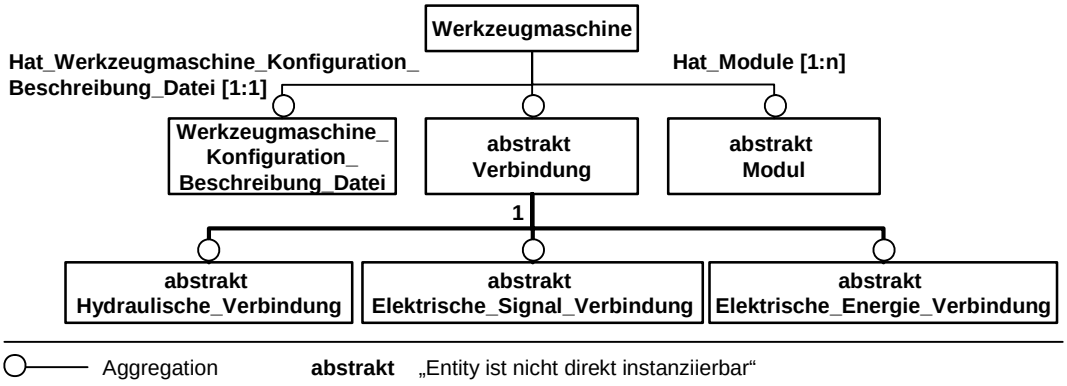


Abbildung 3-6: Ausschnitt aus der Referenzarchitektur einer Werkzeugmaschine in EXPRESS-G [NEUHAUS 2003]

Die Konfiguration einer Werkzeugmaschine mit Hilfe des Konfigurierungsmanagers erfolgt in drei Schritten. Zu Beginn wird eine Analyse der vorhandenen Maschinen- und Steuerungskonfiguration durchgeführt, in einem nächsten Schritt wird die neue Funktionalität parametriert und abschließend wird ein Test durchgeführt. Voraussetzung hierfür sind Konfigurationsdateien, welche auf der Struktur der Referenzarchitektur basieren und somit vom Konfigurationsmanager interpretiert werden können.

Rekonfigurierbare Werkzeugmaschinen ermöglichen eine kostengünstige und flexible Möglichkeit, Produktionslinien an erforderliche Kapazitäten anzupassen bzw. neue Prozessfunktionalitäten zu integrieren [LIAN ET AL. 2000]. In diesem Zusammenhang wird in [KATZ & MOON 2000] eine Architektur für rekonfigurierbare Werkzeugmaschinen vorgestellt sowie das Vorgehen bei der Konfigurierung erläutert. Den Ausgangspunkt stellt hierbei das Teilespektrum dar, das es zu fertigen gilt. Die für die Werkzeugmaschine erforderlichen Funktionen werden dabei von den Anforderungen für die Fertigung der Bauteile abgeleitet.

Eine Analyse der notwendigen Fertigungsoperationen legt die prinzipiellen Rahmenbedingungen der Maschinenkinematik fest. Die Festlegung der Maschinenkonfiguration erfolgt mit dem Softwarewerkzeug *PREMADE*. Das Konfigurationstool erlaubt die graphische Abbildung der Fertigungsoperationen und unterstützt den Anwender bei der Selektion geeigneter Baugruppen-Module. Diese sind in einer Bibliothek in PREMADE hinterlegt. Die Modulbeschreibungen sind mit charakterisierenden Attributen versehen, die PREMADE interpretieren kann [MOON & KOTA 1999]. Auf der Grundlage dieser Attribute und der modellierten Fertigungsoperationen kann PREMADE dem Anwender verschiedene adäquate Maschinenkonfigurationen vorschlagen. Obwohl ausdrücklich erwähnt wird, dass die Modularisierung der Baugruppen in PREMADE auch die Steuerungssoftware sowie die Aktoren und Sensoren umfasst, beziehen sich die Erläuterungen vorwiegend auf die Konfigurierung der mechanischen Funktionen der Werkzeugmaschine. Mit der Erstellung und Modellierung der Software für Maschinensteuerungen befasst sich der folgende Abschnitt.

3.4.3 Entwicklung der Steuerungssoftware

Im Zusammenhang mit der Maschinenkonfigurierung stellt die Softwareerstellung eine besondere Herausforderung dar. Softwarestrukturen sind nicht kongruent zur Maschinen- und Anlagenstruktur aufgebaut. Entsprechende Baukastensysteme für speicherprogrammierbare Steuerungen (SPS) erfordern deshalb andere Beschreibungsmittel und Herangehensweisen. In der industriellen Praxis haben sich die Programmiersprachen der IEC 61131-3 (siehe Anhang 11.4) weitgehend durchgesetzt und genießen eine große Akzeptanz. Allerdings ist die Norm stark implementierungsorientiert festgelegt worden und bietet keine besonderen Möglichkeiten, um beispielsweise Anforderungen an die Steuerungssoftware zu spezifizieren. Durch die steigende Komplexität nimmt der Bedarf nach einem stärker strukturierten Entwicklungsprozess bei der Entwicklung von Steuerungssoftware zu. Insbesondere die Anforderungsspezifikation und das Softwaredesign im Vorfeld der eigentlichen Programmierung erfordern neue Konzepte und Ansätze [FISCHER & VOGEL-HEUSER 2002]. Forschungsansätze, die sich mit der Entwicklung von Steuerungssoftware befassen, setzen deshalb meist auf iterative Konzepte, die eine schrittweise Detaillierung graphischer Softwaremodelle propagieren. Entsprechende Vorgehensweisen und Methoden sind in den Arbeiten von [FLECKENSTEIN 1987], [SCHELBERG 1994] und [MEIER 2001] zu finden.

In [LUTZ 1999] wird eine methodische Unterstützung zur Neu- und Variantenerstellung sowie zur Wiederverwendung von Steuerungssoftware erläutert und in dem Softwarewerkzeug *Aspect* umgesetzt. Ziel ist es, Grundlagen für die ingenieurmäßige Implementierung maschinennaher Steuerungssoftware zu schaffen. Ähnlich wie in [BÖGER 1998] wird ein Informationsmodell definiert, mit dessen Unterstützung Produktionssysteme funktional gegliedert werden und der Steuerungscode objektorientiert modelliert wird. Das Informationsmodell gliedert sich in ein Anlagenmodell und ein Steuerungsmodell (siehe Abbildung 3-7).

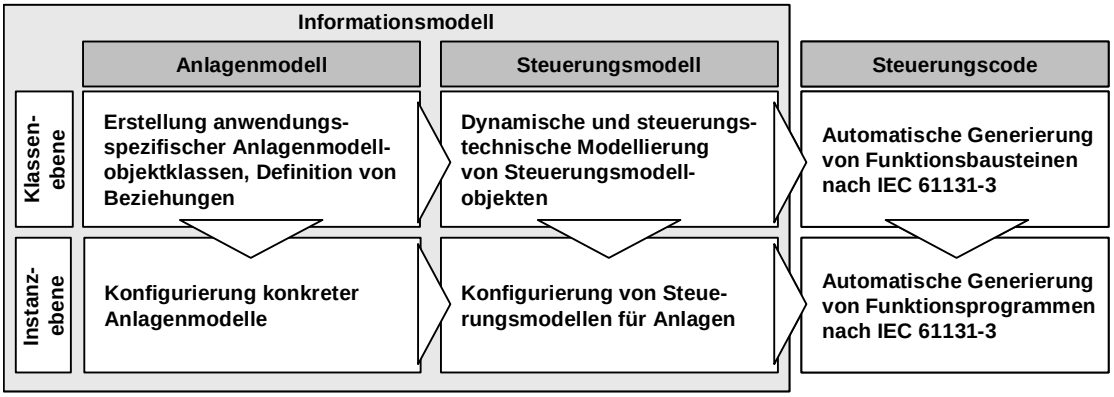


Abbildung 3-7: Informationsmodell und Vorgehensweise nach [LUTZ 1999]

Im Anlagenmodell werden Anlagenmodellobjekte zur Abbildung der Funktionsgliederung und zur Anlagenkonfiguration beschrieben. Im Steuerungsmodell wird das steuerungstechnische Verhalten der Anlagenmodellobjekte dokumentiert. Dabei gilt es zwischen einer Klassen- und Instanzebene zu unterscheiden. Anlagen- und Steuerungsmodellobjektklassen sowie deren Beziehungen zueinander werden auf Klassenebene in einem Metamodell definiert. Den anwendungsspezifischen Anlagenmodellobjektklassen sind geeignete Steuerungsmodellobjekte zugeordnet. Das Metamodell beschreibt Grundstrukturen für den Aufbau anwenderspezifischer Anlagen- und Steuerungsmodelle, welche auf Instanzebene für die Maschinenkonfigurierung verwendet werden. Das Informationsmodell ist die Grundlage von Aspect und dient unter anderem zur Generierung von Steuerungscode.

Die Verbesserung der Steuerungsentwicklung mit einer Virtual-Reality-Umgebung wird in [OSMERS 1998] vorgeschlagen. Schwerpunkt der Arbeit ist die Erstellung fehlerfreier, lauffähiger SPS-Programme noch vor dem Aufbau der eigentlichen Anlage. Der Einsatz von VR-Methoden soll vor allem das räumliche und funktionale Verständnis der Maschineneigenschaften fördern und als Brückenschlag zwischen Mechanikkonstruktion und Steuerungsentwicklung dienen. Es wird das Konzept eines Projektierungswerkzeuges vorgestellt, das Funktionen für die Konfigurierung der Anlage im virtuellen Raum und die Programmierung der Steuerung an diesem Modell ermöglicht. Den Kern des Konzeptes stellt die graphische, interaktive Programmierung von Aktionen und Weiterschaltbedingungen durch einfache Benutzerdialoge dar. Sie basiert auf der visuellen Repräsentation geometrischer Objekte und Steuerungslogikelementen. Voraussetzung für die graphische Programmierung und Konfigurierung ist ein Datenmodell, in dem die einzelnen Objekte definiert sowie ihre Abhängigkeiten abgebildet sind. Darauf aufbauend werden graphische Primitive zur Darstellung geometrischer und logischer Elemente in einer Bibliothek zusammengefasst, aus welcher schließlich das Anlagenmodell zusammengesetzt wird.

Dem Themenbereich störungstoleranter Steuerungen haben sich die Arbeiten von [WAGNER 1997] und [SABBAH 2000] verschrieben. Die in [FISCHER & VOGEL-HEUSER 2002] angesprochene steigende Komplexität von Steuerungsprogrammen führt zu einer sinkenden Verfügbarkeit der Maschinen [MILBERG & EBNER 1994]. Nach [WAGNER 1997] steht dieses Phänomen in Zusammenhang mit der zunehmenden Anzahl an verketteten Komponenten und der Zeit, die für die Fehlerbehebung an der Maschine aufzuwenden ist. Um diesem Sachverhalt entgegenzutreten, wird in der Arbeit ein Konzept zur Fehlererkennung, Fehlerlokalisie-

rung, Fehlerbehebung und zum Wiederanlauf (*Reentry*) vorgestellt. Die Fehlererkennung basiert auf der Prozess- und Zustandüberwachung einzelner *Elementaraktionen*. Diese sind aus Sicht der Maschinensteuerung nicht weiter zerlegbare Funktionen, wie beispielsweise das Schließen eines Werkzeuggreifers. Die Fehlererkennung wird nach dem Zeitpunkt der Erfassung in Pre-Process-, In-Process- und Post-Process-Erkennung unterteilt. Der Zustand der Maschine wird anhand geeigneter Signalgeber erfasst und in einer Matrix gespeichert. Damit die Steuerung selbstständig die Fehlerbehandlung initiieren und ausführen kann, wird das Wissen bezüglich der funktionalen Abhängigkeiten von Elementarfunktionen und des aktuellen Zustands der Maschine in einem Maschinenmodell gespeichert. In Abbildung 3-8 ist ein Ausschnitt davon dargestellt.

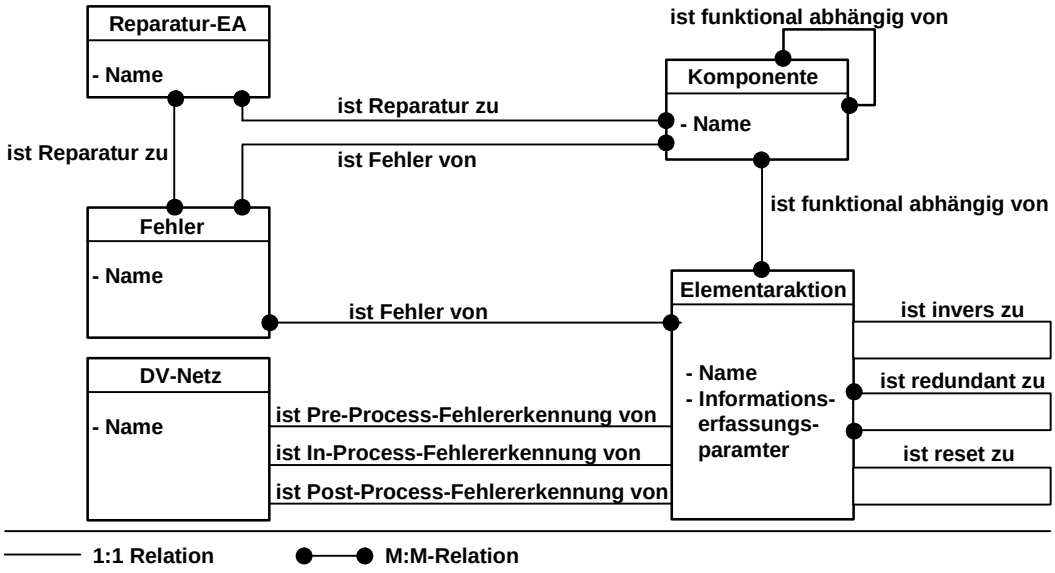


Abbildung 3-8: Ausschnitt aus dem Maschinenmodell nach [WAGNER 1997]

Jede Komponente ist funktional von einer oder mehreren Elementaraktionen abhängig. Auftretende Fehler oder Fehlerbehebungsmaßnahmen (*Reparatur-EA*) werden eindeutig den entsprechenden *Komponenten* oder *Elementaraktionen* zugeordnet. Die Informationsverarbeitung über den aktuellen Zustand der Maschine erfolgt in dem Konstrukt *DV-Netz*. Hier wird die Matrix der aktuellen Prozess- und Zustandsinformationen verwaltet. Individuelle, auf die Elementaraktionen abgestimmte, Mechanismen zur Pre-, In- und Post-Process-Fehlererkennung verarbeiten diese Informationen und generieren daraus Aussagen zur Fehlerfreiheit bzw. Fehlerhaftigkeit der Elementaraktionen. Die Fehlerlokalisierung sowie die Fehlerbehebung erfolgt durch die im Maschinenmodell abgebildeten Abhängigkeiten zwischen den Elementaraktionen, dem Fehler, der Komponente und der Reparatur-EA. Das vorgestellte Konzept dient als Vorlage für die Definition einer Steuerungsstruktur mit integrierter Fehlerbehandlung. Als Anwendungsbeispiel wurde die SPS-Steuerung eines Bearbeitungszentrums durch die erarbeitete Steuerungsstruktur ersetzt und deren Praxistauglichkeit nachgewiesen.

Die Arbeit von [SABBAH 2000] legt den Schwerpunkt auf die methodische Entwicklung störungstoleranter Steuerungen. Die Methode unterstützt dabei alle Phasen der Störungsbehand-

lung und bietet modellbasierte Beschreibungsmittel sowie eine entsprechende Entwicklungsumgebung an. Kern des vorgestellten Referenzkonzepts ist der störungstolerante Steuerungsbaustein. Dieser Baustein bietet so genannte Dienste zur Steuerung einzelner Komponenten oder Baugruppen an. Die Steuerungsfunktionalität wird mittels Zustandsgraphen der UML abgebildet. Darüber hinaus enthält jeder Steuerungsbaustein einen Fehlerbaum, der für die Fehlerbehandlung eingesetzt wird. Der Fehlerbaum verfügt über Mechanismen zur Erfassung und Manipulation von Steuerungssignalen sowie über Funktionen, mit denen standardisierte Bedienerdialoge geführt werden können.

Den besonderen Herausforderungen der Softwareentwicklung im Automatisierungsbereich widmet sich auch die Arbeit von [HEVERHAGEN 2003]. Hier wird ein Konzept vorgestellt, das die Ebene der Prozesssteuerung mit der Ebene der Betriebsdatenerfassung, Produktionsplanung und Produktionssteuerung zusammenführt. Im Fokus liegt hierbei die Konzeption und Implementierung geeigneter Schnittstellen, der so genannten *Funktionsbausteinadapter*. Die Programmierung von speicherprogrammierbaren Steuerungen erfolgt hauptsächlich durch Sprachen der IEC 61131-3. In der Entwurfsphase werden in erster Linie Funktionsbausteine und die Ablaufsprache verwendet. Dem Konzept einer Programmorganisationseinheit entspricht dabei der Funktionsbaustein. Die PC-basierten Automatisierungsgeräte höherer Ebene werden hingegen mit objektorientierten Sprachen programmiert und mit der UML konzipiert. Anhand der Funktionsbausteinadapter können bereits in der Entwurfsphase eines Systems Kommunikationsbeziehungen zwischen den Funktionsbausteinen der IEC 61131-3 und der UML-Modelle spezifiziert werden.

3.4.4 Blockschaltbild zur Beschreibung des Maschinenverhaltens

Nicht nur die Logik der SPS lässt sich als Modell beschreiben, sondern auch das Verhalten der Hardwarekomponenten kann mithilfe von so genannten *Blockschaltbildern* spezifiziert werden. Diese stellen verschiedene Zeitglieder und mathematische Funktionen zur Abbildung der Maschineneigenschaften zur Verfügung. Im Folgenden werden zwei unterschiedliche Möglichkeiten für die Darstellung von Blockschaltbildern vorgestellt.

MATLAB/Simulink

MATLAB/Simulink ist eine proprietäre Entwicklungsumgebung und gehört zu den blockschaltbildorientierten Systemen [HOFFMANN 1998]. Sie wird für den Entwurf, für die Simulation und für die Bewertung dynamischer Systeme eingesetzt. MATLAB/Simulink setzt sich aus verschiedenen so genannten *Toolboxen* zusammen. Zurzeit existieren mehr als 150 verschiedene Toolboxen für spezifische Anwendungsbereiche wie die Steuerungs- und Regelungstechnik, die Betriebswirtschaft oder die Statistik. Simulink ist ebenfalls eine Toolbox und stellt die graphische Entwicklungs- und Simulationsumgebung dar. Unter MATLAB wird die numerische Programmierumgebung verstanden, die die Programmierung und Berechnung der mathematischen Systemgleichungen ermöglicht.

Das Verfahren zur Modellbildung basiert auf der Beschreibung von Signalflüssen und ist somit für die Abbildung regelungstechnischer Systeme geeignet. Die einzelnen Modellelemente werden als *Blöcke* bezeichnet. Diese repräsentieren die Systemeigenschaften auf der Grundla-

ge formalisierter mathematischer Gleichungen und besitzen Ein- und Ausgangsgrößen. Die Blöcke werden im Blockschaltbild entlang einer Wirkungsrichtung miteinander verschaltet, wobei mit den mathematischen Systemgleichungen aus den Eingangsgrößen die entsprechenden Ausgangsgrößen berechnet werden. In diesem Zusammenhang spricht man auch von *kausaler Modellierung*, da die Signalrichtung die Auswertungsvorschrift festlegt [GRÄB 2001]. Ein Block ist entweder ein Basisblock mit einer definierten mathematischen Funktion oder er fasst weitere, miteinander verbundene Blöcke zu einer hierarchischen Struktur zusammen. Durch dieses Hierarchisierungskonzept ist es möglich, das mathematische Modell eines mechanischen Systems beispielsweise als Ergebnis einer *Mehrköpersimulation* (MKS) oder *FEM-Analyse* zu importieren [ZÄH ET AL. 2002]. Die einzelnen Blöcke werden in MATLAB/Simulink in Bibliotheken zusammengefasst und verwaltet. In Abbildung 3-9 ist das Beispiel einer hydraulischen Wegregelung einer Reibschweißanlage dargestellt.

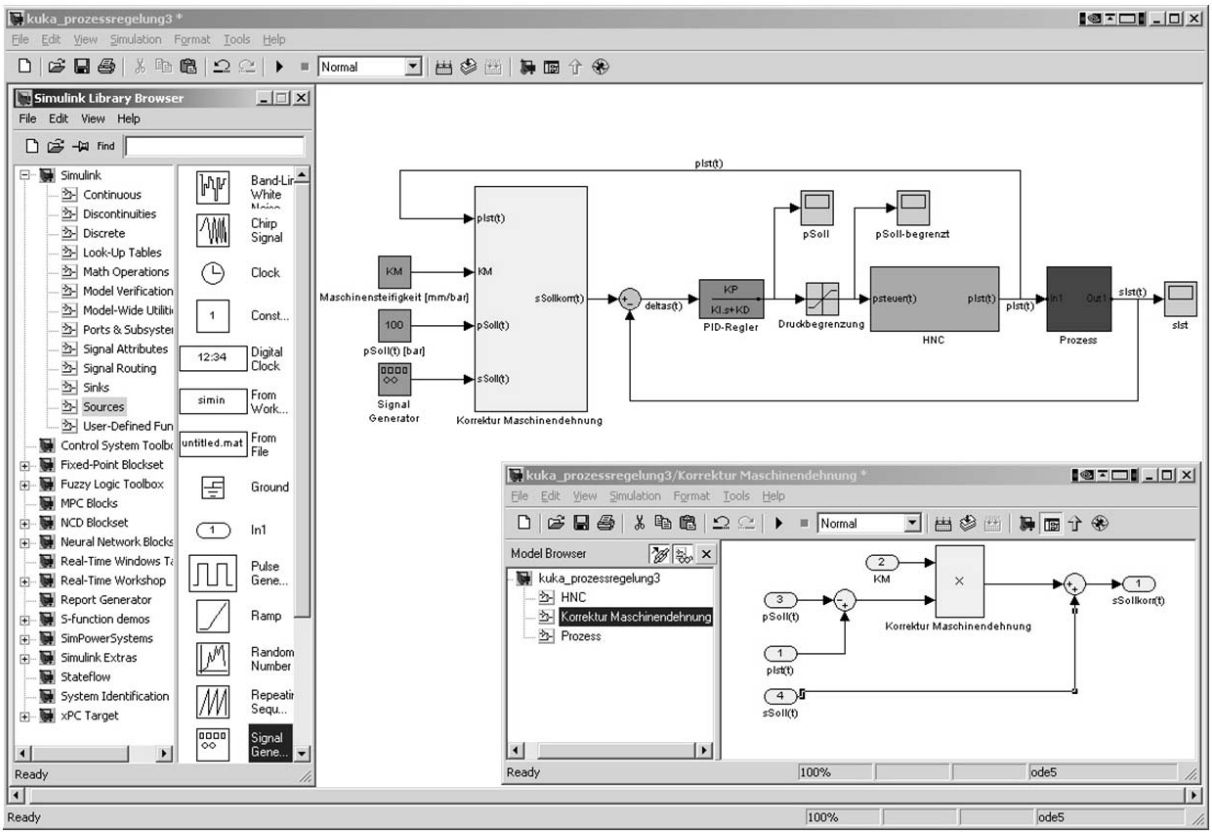


Abbildung 3-9: MATLAB/Simulink-Modell

Da sich die Anlage durch die hohen Prozesskräfte ausdehnt, muss der Sollwert um die Maschinendehnung korrigiert werden. Anschließend wird der Lagesollwert mit dem Istwert verglichen und die Regelabweichung berechnet. Der nachfolgende *PID-Regler* berechnet den Druck-Sollwert zum Ausgleichen der Wegdifferenz. Ein höherer Druck bedeutet hierbei eine schnellere Verfahrbewegung und damit eine Reduktion der Regelabweichung. Danach erfolgt eine Solldruckbegrenzung, da beim Schweißen nur eine gewisse Schwankungsbreite erlaubt ist. Der begrenzte Druck-Sollwert *pSollbegrenzt* wird an die hydraulische Steuerung (HNC) übergeben, die auf der Basis des vorgegebenen, begrenzten Solldruckes die Druckregelung übernimmt. Über den Schweißprozess, der als Block auf der Basis von Versuchsergebnissen

modelliert worden ist, ergibt sich ein Verfahrweg *SIst*. Dieser wird wieder an die Regelung zurückgeführt. Die Bildschirme dienen zur Darstellung der Simulationsergebnisse. Das Subsystem *Korrektur Maschinendehnung* ist in einem gesonderten Fenster detailliert dargestellt.

Modelica

Modelica ist eine frei verfügbare Modellierungssprache zur Beschreibung und Simulation mechatronischer Systeme. Sie wird seit 1996 von der gemeinnützigen *Modelica Association* weiterentwickelt und basiert in ihren Grundzügen auf den Arbeiten von Hilding Elmqvist [ELMQVIST 1978]. Im Gegensatz zu MATLAB/Simulink verfolgt Modelica einen objektorientierten Ansatz, bei dem die topologische Struktur der Komponenten des mechatronischen Systems direkt auf die Modellierungselemente der Sprache abgebildet wird. Dabei werden keine kausalen Zusammenhänge zwischen den einzelnen Komponenten beschrieben, sondern ausschließlich die Beziehungen definiert. Die Schnittstellen können aus gerichteten Signalen bestehen aber auch physikalische Verbindungen darstellen, wie beispielsweise einen mechanischen Flansch oder eine elektrische Klemme.

Modelica ist eine objektorientierte Sprache und weist in diesem Zusammenhang zahlreiche Parallelen zu objektorientierten Programmiersprachen wie C++ oder Java auf, die sich allerdings im Detail unterscheiden können. Auf der Anwenderebene werden Modelica-Modelle durch Objektdiagramme beschrieben. Ein Objektdiagramm ist die Verallgemeinerung eines Blockschaltbildes, in dem die einzelnen Modellelemente durch geeignete Symbole dargestellt werden. Die Modellelemente eines Objektdiagramms sind wiederum hierarchisch aus Objektdiagrammen aufgebaut. Auf der untersten Ebene werden die Modellelemente durch Differential-, algebraische und diskrete Gleichungen beschrieben (siehe Abbildung 3-10).

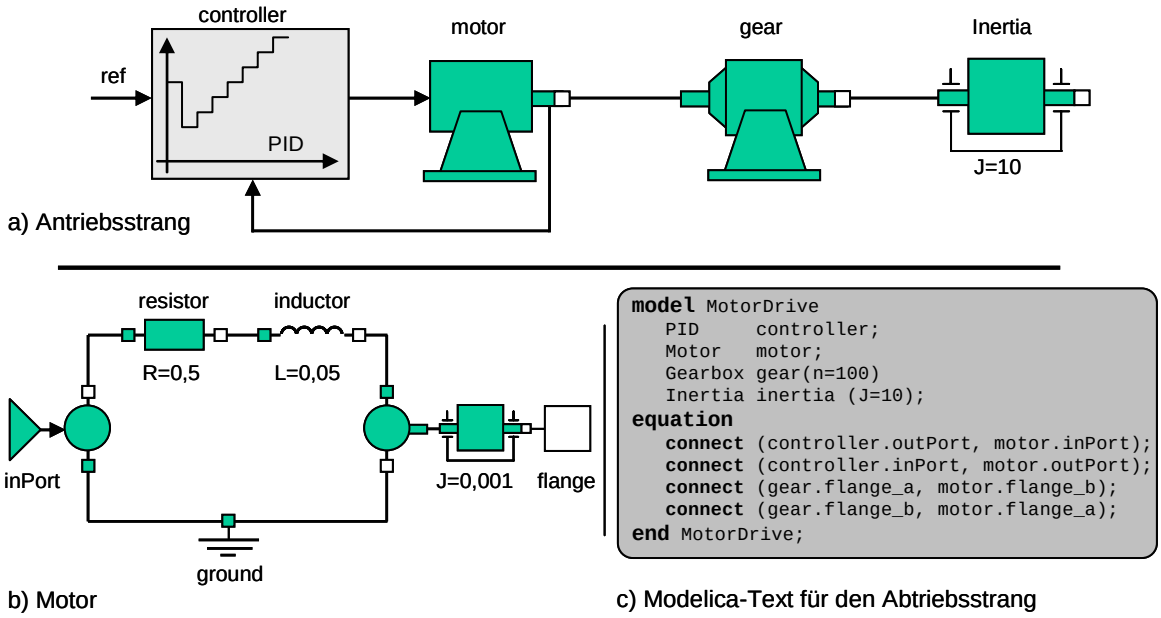


Abbildung 3-10: Objektdiagramm eines Antriebsstrangs [OTTER & SCHWEIGER 2005]

Das grundlegende Modellelement von Modelica ist die *Klasse*. Klassendefinitionen werden mit den Schlüsselworten *class* oder *model* eingeleitet und beschreiben die Strukturen einzel-

ner Komponenten oder ganzer Systeme. In den Klassen können Variablen, Konstanten und mathematische Gleichungssysteme spezifiziert werden, wobei letztere das Verhalten der Instanzen der jeweiligen Klasse festlegen. Die hierfür notwendigen mathematischen Gleichungssysteme werden im *equation*-Bereich der Klassen angegeben. Instanzen werden in Modelica mit dem Schlüsselwort *component* bezeichnet. Diese können über Schnittstellen miteinander verbunden werden und somit Daten austauschen. Die Schnittstellen werden mit dem Schlüsselwort *connector* bezeichnet und werden in den *connector-classes* definiert. Zur Verdeutlichung ist in Abbildung 3-10 a ein Modell eines Antriebsstrangs zu sehen.

Das Objektdiagramm in Abbildung 3-10 a zeigt den gesamten Antriebsstrang mit dem Regler, dem Motor, dem Getriebe und einer Last. Die textorientierte Beschreibung ist in Abbildung 3-10 c in der Modellklasse *MotorDrive* zu sehen. Zu Beginn werden die vier Komponenten instanziiert. Die Anweisung *Gearbox gear (n=100)* erzeugt eine Komponente *gear* der Klasse *Gearbox* und initialisiert den Parameter *n* mit dem Wert *100*. Der Parameter n ist eine Variable, die in der Klasse Gearbox deklariert ist. Im *equation*-Bereich der Klasse ist die Verschaltung der Schnittstellen der einzelnen Komponenten definiert. Mit dem Befehl *motor.inPort* wird beispielsweise auf den *Connector inPort* der Klasse *Motor* zugegriffen und durch die Anweisung *connect* mit dem Connector *controller.outPort* verschaltet. Das Objektdiagramm der Klasse *Motor* ist der Abbildung 3-10 b zu entnehmen. Das Dreieck stellt den *Connector inPort* als Signaleingang dar, wohingegen das Viereck eine Schnittstelle als mechanischen Flansch abbildet. Modelica stellt in den verschiedenen Bibliotheken eine Vielzahl elementarer *Connectoren* zur Verfügung, aus denen anwendungsspezifisch hierarchische *Connectoren* aufgebaut werden können. Um die Klassen in den einzelnen Bibliotheken verwalten und strukturieren zu können, besitzt Modelica ein Modularisierungskonzept, das die Klassen einem Paket zuordnet. Klassen können dann bei Bedarf durch eine *import*-Anweisung in andere Pakete referenziert werden.

Bevor ein Modell simuliert werden kann, muss aus der objektorientierten Struktur ein System von Differential-algebraischen Gleichungen abgeleitet werden. Diese Aufgabe übernimmt ein Modelica-Übersetzer wie beispielsweise *Dymola*, indem er die Gleichungen aller Komponenten der Objektdiagramme und Gleichungen aufgrund von connect-Verbindungen zu einem Gleichungssystem zusammenfasst. Nähere Informationen zu den hierzu verwendeten Transformationsalgorithmen können [ELMQVIST 1978], [RICHERT ET AL. 2003] und [OTTER & SCHWEIGER 2005] entnommen werden. Detaillierte Informationen über die einzelnen Sprachelemente und den Aufbau von Modelica-Modellen sind in [FRITZSON 2004] zu finden.

3.4.5 Zusammenfassung

Die aufgeführten Arbeiten zeigen den Einsatz unterschiedlicher Modellkonzepte in den verschiedenen Phasen der Systementwicklung. Von großer Bedeutung ist dabei die Beschreibung von Funktionsstrukturen in den frühen Phasen der Entwicklung. Diese fördern insbesondere das interdisziplinäre Problemverständnis und bilden die Handlungs- und Entscheidungsgrundlage der beteiligten Fachbereiche. Dies wird dadurch ersichtlich, dass nahezu jeder vorgestellte Ansatz den Begriff *Funktion* oder ein semantisches Äquivalent als eigenständiges Modellelement oder Sprachkonstrukt definiert und in den jeweiligen anwendungsspezifischen Kontext integriert. Dieser anwendungsspezifische Kontext wird anhand geeigneter Notationen

formalisiert dargestellt und liefert die Grundlage für die Daten- und Informationsmodelle der entwickelten Softwareprototypen.

3.5 Entwicklungsumgebungen und Simulationssysteme

Während in den vorhergehenden Abschnitten der Schwerpunkt auf Methoden und Modellierungstechniken lag, werden im Folgenden spezifische Entwicklungsumgebungen für verschiedene Bereiche der Werkzeugmaschinenentwicklung erläutert. Die in der Mechanikkonstruktion eingesetzten CAD-Werkzeuge werden dabei nicht explizit behandelt. Für nähere Informationen diesbezüglich sei auf [SPUR & KRAUSE 1997] verwiesen. Im Rahmen der vorliegenden Arbeit wird angenommen, dass die in der Mechanikkonstruktion anfallenden Aufgaben mit einer featurebasierten 3D-CAD-Entwicklungsumgebung ausgeführt werden.

3.5.1 Planung elektrischer und fluidtechnischer Betriebsmittel

Beim Betrieb von Werkzeugmaschinen laufen auf physikalischer Ebene unterschiedliche Prozesse ab, beispielsweise Zerspan-, Gleit-, Roll-, Umformungs- oder Fließprozesse. Bei der technologischen Bearbeitung des Werkstücks sind eine Vielzahl an Bauteilen und Baugruppen, wie beispielsweise Antriebe, Pumpen oder Werkzeug- und Palettenwechsler, beteiligt. Die Vielfalt an unterschiedlichen Prozessen in Werkzeugmaschinen ist dementsprechend hoch. Jeder dieser Prozesse induziert physikalische Wechselwirkungen mit dem System, die durch geeignete Sensoren erfasst werden können. Auf der Grundlage der gelieferten Signalwerte beeinflusst die Maschinensteuerung diese Prozesse, indem sie Steuersignale an Aktoren und Stellglieder ausgibt [WAGNER 1997].

In diesem Zusammenhang spricht man auch von elektrischen, hydraulischen und pneumatischen *Betriebsmitteln* oder *Feldkomponenten* und meint damit Sensoren, Aktoren, Stellglieder sowie den Signal- und Energietransport im Maschinenfeld. Die Hydraulik dient vorwiegend als binärer Energieträger für Spann- und Klemmvorrichtungen, ist in Ausnahmefällen allerdings auch in Form geregelter hydraulischer Antriebe zu finden. Pneumatische Systeme fungieren als Medium für Sperrluft bzw. für Reinigungsfunktionen. In manchen Bereichen ersetzt die Pneumatik auch die Hydraulik, vor allem in so genannten Peripheriefunktionen wie beispielsweise dem Werkzeugwechsler [WAGNER 1995]. Mit der *dezentralen und standardisierten Installationstechnik an Werkzeugmaschinen und Produktionsanlagen (DESINA)* wurde eine firmenübergreifende Vereinheitlichung der Betriebsmittel erreicht. Anfang November 2005 wurde DESINA als internationale Norm ISO DIS 23570 [ISO DIS 23570 2005] anerkannt. DESINA legt Komponentenspezifikationen und Projektierungsrichtlinien fest, auf deren Basis die kostentreibende und oftmals verfügbarkeitsmindernde Feldinstallation von Produktionsmaschinen und -anlagen optimiert werden kann.

Die Aufgaben bei der Betriebsmittelplanung liegen vor allem in der Auswahl geeigneter Komponenten und deren Integration in das Maschinenfeld. Die Planung der Installationsstruktur wird von der Elektrokonstruktion übernommen, nachdem die Mechanikkonstruktion die grundlegende Topologie der Maschine bereits festgelegt hat. In Abbildung 3-11 sind zwei zentrale Modellkonzepte bei der Festlegung der Installationstechnik dargestellt.

Die Beschreibung hydraulischer und pneumatischer Komponenten verhält sich analog und wird deshalb nicht gesondert erläutert.

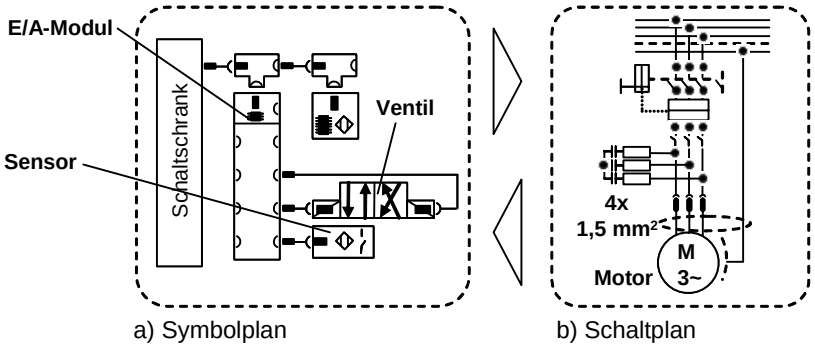


Abbildung 3-11: Modellbeschreibungen bei der Betriebsmittelplanung [WAGNER 1995]

Sind die Betriebsmittel im Maschinenfeld platziert, wird der Bedarf an Verbindungen geklärt. Dabei sind Rahmenbedingungen wie Kabellängen oder der Verlegeort zu beachten. Grundlage für die Verlegung von Leitungen sind Symbolpläne (Abbildung 3-11 a). Sie werden verwendet, um die Anordnung und äußere Verdrahtung von Betriebsmitteln darzustellen. Die Abbildung erfolgt in einer lagerichtigen, jedoch nicht maßstäblichen, einpoligen Notationsform und basiert auf den Normen DIN EN 60617 (Graphische Symbole für Schaltpläne) [DIN EN 60617 1997] sowie DIN ISO 1219 (Fluidtechnik - Graphische Symbole und Schaltpläne) [DIN ISO 1219 2004]. Softwarewerkzeuge zur Erstellung von Symbolplänen ermöglichen meist eine dynamische Anpassung des Modells. Ändern sich mechanische Grundstrukturen, werden vom Softwarewerkzeug automatisch alle Referenzen zu den Betriebsmitteln modifiziert.

Die Schaltpläne [DIN EN 61082 1996] dienen zur Illustration der Verschaltung der Betriebsmittel und werden mittels ECAD-Systemen erstellt (Abbildung 3-11 b). Sie bieten eine detaillierte Darstellung der einzelnen Stromwege der Betriebsmittel und sind wesentlich detaillierter als Symbolpläne. Derartige Systeme besitzen umfangreiche und erweiterbare Symbolbibliotheken und unterstützen den Anwender bei der Schützauswahl, bei der Erstellung von Klemmenanschluss- und Kabelplänen und bei der Stücklistengenerierung und der Kabelkonfektionierung. Moderne ECAD-Systeme bezeichnet man auch als Systeme der dritten Generation und meint damit vor allem die Unterstützung von Variantenkonstruktionen, auf nationalen und internationalen Standards basierende Schnittstellen sowie eine Datenverwaltung mit objektorientierten oder relationalen Datenbanken [ROLLER & SCHÄFER 2002]. Moderne ECAD-Systeme bieten Schnittstellen zum Anwendungsprotokoll AP 212 der ISO 10303-212 [ISO 10303-212 2004]. Das AP 212 ermöglicht die Datendurchgängigkeit in den Prozessketten der Elektrotechnik und bietet Lösungen, um die in Schaltplänen oder Stücklisten enthaltenen Daten herstellerunabhängig beschreiben zu können. Das AP 212 berücksichtigt die elektrischen und mechanischen Eigenschaften eines elektrischen Produktes und gestattet damit die vollständige Beschreibung der Schaltungslogik sowie aller zugehörigen technischen und nichttechnischen Daten [UNGERER & FISCHER 2002]. Allerdings unterstützt nur ein geringer Anteil der am Markt verfügbaren ECAD-Systeme diese Norm, so dass ein direktes Verarbeiten der auf AP 212 basierenden Datenformate nicht möglich ist [ROLLER & SCHÄFER 2002].

Ein innovativer, neuer Ansatz für die Elektrokonstruktion wurde in dem Forschungsprojekt *Föderal* entwickelt [FÖDERAL 2004]. Ziel war es, ein methodisches Vorgehen und ein modulares Engineeringsystem für die interdisziplinäre Projektierung von Maschinen und Anlagen zu erarbeiten. Das Föderal-System besteht aus einer Kommunikationsplattform, der so genannten *Föderalen Informations-Architektur* (FIA). An diese lassen sich bereits vorhandene Softwarewerkzeuge in Form einzelner Module anbinden, beispielsweise ECAD-Systeme oder SPS-Programmierumgebungen. Den Daten- und Informationsaustausch zwischen den Softwarewerkzeugen übernimmt die FIA auf der Grundlage semistrukturierter Datenmodelle [MANGOLD ET AL. 2003]. Dadurch sind eine disziplinübergreifende Projektierung von Maschinen und Anlagen sowie die Generierung vollständiger Projektunterlagen, die sich in vorhandene Systeme übernehmen lassen, möglich.

Ausgangspunkt der im Projekt Föderal erarbeiteten Vorgehensweise ist das funktionale Engineering auf der Grundlage *virtueller mechatronischer Komponenten*. Eine definierte Maschinenfunktion besteht in ihrer Summe aus einer Kombination von Teilkomponenten unterschiedlicher Disziplinen. Eine virtuelle mechatronische Komponente fasst diese Teilkomponenten nach funktionalen Kriterien zusammen. In Abbildung 3-12 ist das Konzept der funktionalen Projektierung dargestellt.

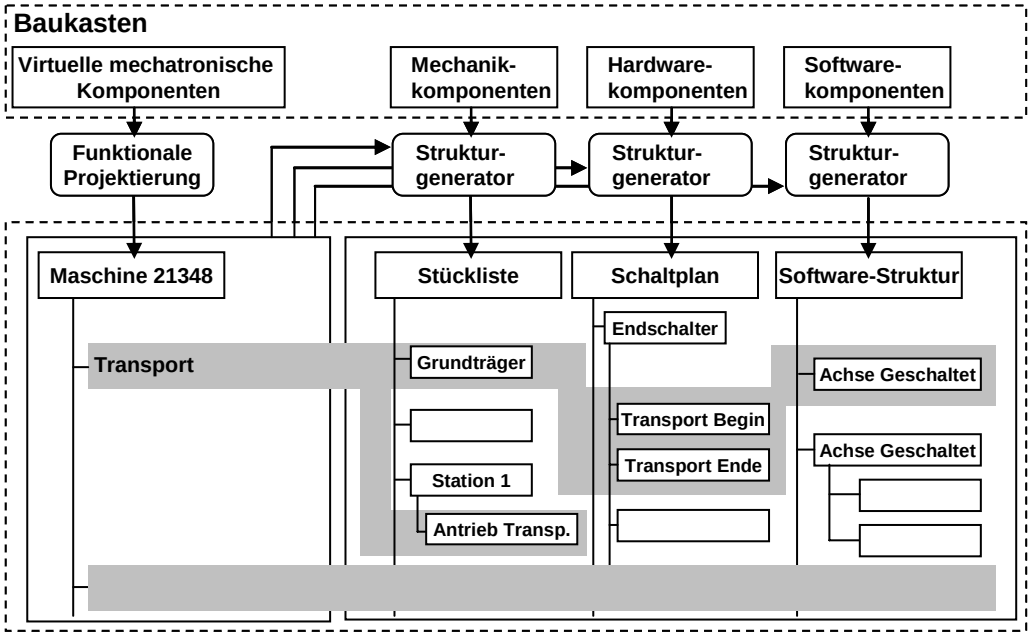


Abbildung 3-12: Funktionale Projektierung mit Föderal [FÖDERAL 2004]

In der FIA werden die virtuellen mechatronischen Komponenten in einem Baukasten verwaltet. Die Projekteure konfigurieren die Maschine mit Hilfe dieser Komponenten und deren Parameterwerten. Wie das Bild zeigt, beinhalten die Komponenten Referenzen zu den disziplinspezifischen Teilkomponenten. Durch sie wird die Gesamtmenge der Komponenten für jede Disziplin ermittelt und für die Erstellung der disziplinspezifischen Strukturen (beispielsweise Schaltpläne) verwendet. Strukturgeneratoren automatisieren diesen Vorgang, sofern eindeutige Regeln für den Strukturaufbau definiert wurden.

Derartige Strukturgeneratoren verwenden zum Beispiel die Makro-Programmiersprachen der ECAD-Systeme. So besitzt die virtuelle mechatronische Komponente für die Funktion *Spannen* Referenzen zum Schaltplan und einen allgemein gültigen Funktionsbaustein für geschaltete Achsen.

Durch die Parametrierung wird die virtuelle mechatronische Komponente projektspezifischen Anforderungen angepasst. Die Strukturgeneratoren erzeugen daraus die erforderlichen Schaltpläne und den SPS-Code [FÖDERAL 2004].

Das auf der FIA basierende, föderale funktionale Engineering zeigt die Möglichkeiten und das Potenzial der modellgetriebenen Entwicklung auf. Der Schwerpunkt liegt dabei auf der Erstellung der elektrotechnischen Dokumentation und der entsprechenden Vorgehensweise für den Aufbau eines Baukastens zur Konfigurierung von Maschinen und Anlagen. Möglichkeiten zur funktionalen Beschreibung von Maschinenabläufen und deren Simulation waren nicht Gegenstand der Arbeiten.

3.5.2 Abbilden strukturdynamischer Maschineneigenschaften

In der Mechanikkonstruktion werden schon seit längerem durchgängige Modellierungstechniken und Vorgehensweisen zur Analyse und Verifikation von Maschineneigenschaften eingesetzt. Die Aktivitäten in diesem Bereich stützten sich vor allem auf die *Finite-Elemente-Methode* (FEM) und auf die *Mehrkörpersimulation* (MKS). Ausgangspunkt ist hierbei die 3D-Volumenmodellierung anhand geeigneter 3D-CAD-Systeme. Durch die anschließende Anwendung der FEM bzw. MKS können bereits am Rechner grundlegende Fragen bezüglich der Optimierung der mechanischen Auslegung geklärt werden. Im Anschluss werden die beiden Methoden FEM und MKS vorgestellt.

Die Methode der Mehrkörpersimulation (MKS)

Die Methode der Mehrkörpersimulation (MKS) wird für die Abbildung des dynamischen Verhaltens mechanischer Systeme eingesetzt, bei denen die einzelnen Komponenten des Systems große Bewegungen ausführen und deren Verformungen selbst vernachlässigbar gering sind. Damit ist das Strukturverhalten als stark nichtlinear anzunehmen. Die Methode der Mehrkörpersimulation basiert darauf, dass mechanische Eigenschaften wie Trägheit, Elastizität und Viskosität sowie zu übertragende Kräfte einzelnen diskreten Elementen zugeordnet werden. Die Komponenten eines realen Systems werden als so genannte Starrkörper abgebildet, auf die an den Koppelpunkten Einzelkräfte und -momente einwirken. Die Kräfte und Momente werden durch Bindungen und Koppelelemente hervorgerufen [KÜBLER 2000]. In Abbildung 3-13 ist ein Auszug der wichtigsten Modellelemente in der Mehrkörpersimulation zu sehen.

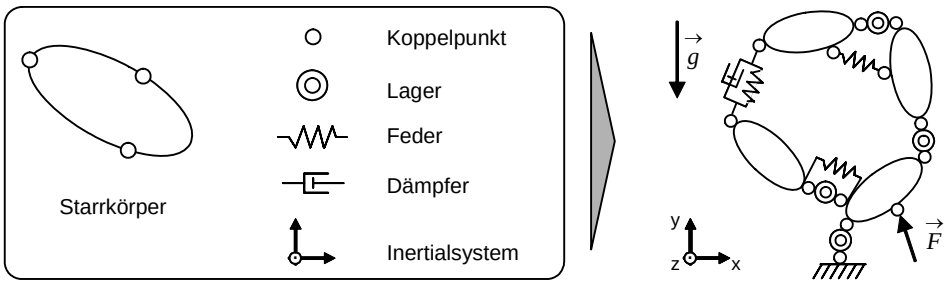


Abbildung 3-13: Modellelemente in der Mehrkörpersimulation [BAUDISCH 2003]

Im Mehrkörpermodell werden den Starrkörpern die Geometrieinformationen sowie die Masse und die Trägheitseigenschaften der Komponenten zugewiesen, die sie repräsentieren. Die Starrkörper werden durch Gelenke miteinander verbunden und können sich somit gegeneinander bewegen. Die Gelenke werden dabei als ideal, d. h. starr und reibungsfrei, angenommen. Zwischen den Körpern wirken Kraftelemente, die Kräfte und Momente in Abhängigkeit ihrer kinematischen Ausprägungen einbringen. Im einfachsten Fall handelt es sich bei den Kraftelementen um Federn und Dämpfer, es lassen sich darüber hinaus auch komplexe Kontaktmodelle durch Kraftmodelle beschreiben [HIPPMANN 2004]. Tiefergehende Informationen zur Methode der Mehrkörpersimulation finden sich in den Lehrbüchern von [SCHIEHLEN 1986], [NIKRAVESH 1988] und [HAUG 1989].

Die Finite-Elemente-Methode (FEM)

Die *Finite-Elemente-Methode* ist ein numerisches Näherungsverfahren für die Lösung partieller Differentialgleichungssysteme. Die prinzipielle Vorgehensweise beruht darauf, die zu berechnende Struktur in eine bestimmte Anzahl einfacher, endlich großer Elemente (Finite Elemente) zu zerlegen, deren Eigenschaften exakt oder näherungsweise bekannt sind [BAUDISCH 2003]. Der Anwender teilt hierzu die Struktur des zu berechnenden Systems in ein Knotenpunktnetz ein [FINKE 1977]. In Abbildung 3-14 ist ein entsprechendes FE-Netz aufgeführt. Um lokale Geometrie- und Stoffeigenschaften möglichst gut beschreiben zu können, sind geeignete Elemente zum Aufbau des FE-Netzes auszuwählen.

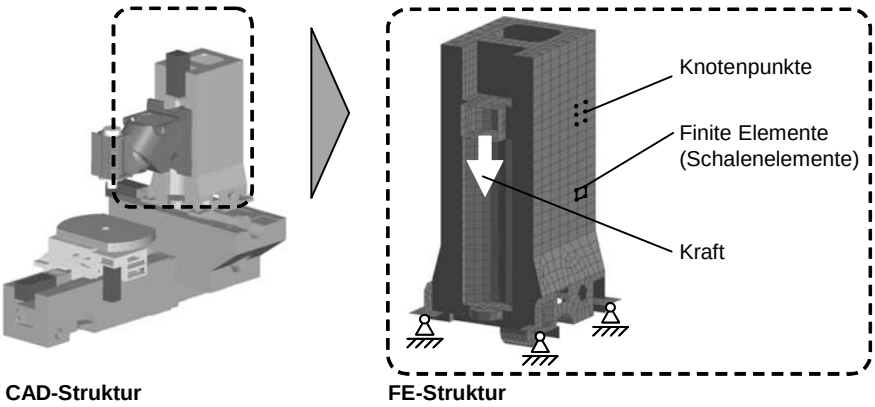


Abbildung 3-14: Knotenpunktnetz bei der Finite-Elemente-Methode

Die Auswahl richtet sich dabei nach dem Spannungszustand im Bauteil bzw. nach den Freiheitsgraden, in denen Verschiebungen oder Verdrehungen an den Knotenpunkten auftreten bzw. Belastungen auf das Element einwirken können. Einachsige Spannungszustände können mit Stabelementen abgebildet werden, während Wellenelemente zusätzlich Torsionsmomente und Balkenelemente Belastungen in allen Freiheitsgraden aufnehmen können. Für einen zweiachsigen Spannungszustand mit zusätzlicher Belastung senkrecht zur Elementebene werden Platten- oder Schalenelemente verwendet [MILBERG 1992]. Typische Anwendungsgebiete der FEM-Berechnung sind die Ermittlung von Bauteilverformungen unter Last sowie die Ermittlung von Eigenfrequenzen mit den zugehörigen Eigenformen [BAUDISCH 2003].

3.5.3 Interdisziplinäre Simulationssysteme

Der Einsatz von fachbereichsübergreifenden Simulationsmodellen in der Werkzeugmaschinenentwicklung ist von immer größerer Bedeutung. Durch die engen Zeit- und Kostenrahmen ist es für die Hersteller kaum möglich, Prototypen für den Maschinentest zu bauen und ein entsprechendes Test-Team zu finanzieren. Aus diesem Grund steigt der Bedarf der Industrie an geeigneten Vorgehensweisen, Modellierungstechniken und Softwarewerkzeugen, um Maschineneigenschaften im Gesamtkontext frühzeitig am Rechner verifizieren und validieren zu können. Dabei bilden unter anderem die FEM und die MKS die Grundlage für weitergehende Konzepte und Forschungsarbeiten, welche im Folgenden aufgeführt werden.

In [ALBERTZ 1995] wurde eine Methode erarbeitet, die detailliert den Ablauf des Entwicklungsprozesses unter dem Einsatz der FEM-Simulation für die Entwicklung dynamikgerechter Gestellstrukturen beschreibt. Die Arbeit von [SCHNEIDER 2000] erweitert diesen Ansatz, indem er die Strukturkomponenten getrennt voneinander durch die Bewertung von verschiedenen Varianten optimiert. Um Lastfälle bei verschiedenen Belastungssituationen zu bestimmen, können die MKS- und FEM-Modelle gekoppelt werden. Hierzu werden FE-Körper in das Mehrkörpermodell integriert, die dann an verschiedene Positionen bewegt werden können. In [REINHART ET AL. 2001C] wird diese Methode benutzt, um das arbeitsraumabhängige Verhalten einer Fräsmaschine abzubilden.

In [NEITHARDT 2004] wird eine Methodik beschrieben, wie das mechanische Verhalten von Werkzeugmaschinen hinsichtlich ihrer Grundstruktur in der Konzeptphase sowie der Bauteilstruktur in der Entwurfsphase arbeitsraumabhängig simuliert und optimiert werden kann. Basis für die Durchgängigkeit ist hierbei die Verwendung eines MKS-Systems.

Die Simulation im steuerungstechnischen Umfeld ist in der Industrie noch nicht sehr verbreitet, da es in diesem Bereich noch keine derart etablierten und anerkannten Modellierungstechniken und Softwarewerkzeuge gibt. In zahlreichen Forschungsansätzen werden jedoch entsprechende Lösungsvorschläge erarbeitet.

Eine Methode zur Optimierung der Entwicklung der Steuerungssoftware von Werkzeugmaschinen wird in [ZAEH & POERNBACHER 2005] präsentiert. Auf der Grundlage der UML wurde eine Modellierungstechnik und eine Vorgehensweise entwickelt, um das steuerungstechnische und physikalische Verhalten der Maschine zu modellieren. In Abbildung 3-15 ist ein Ausschnitt der Strukturbeschreibung eines Werkzeugwechslers zu sehen.

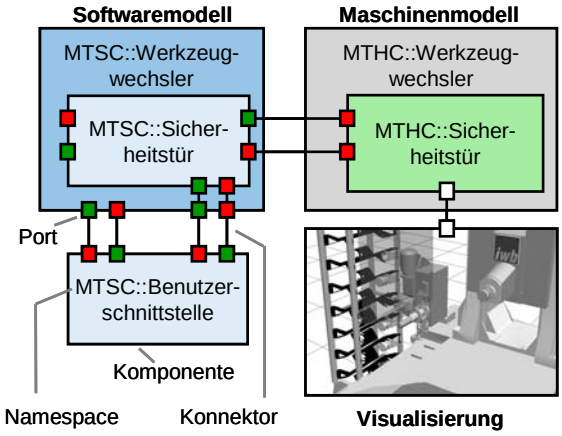


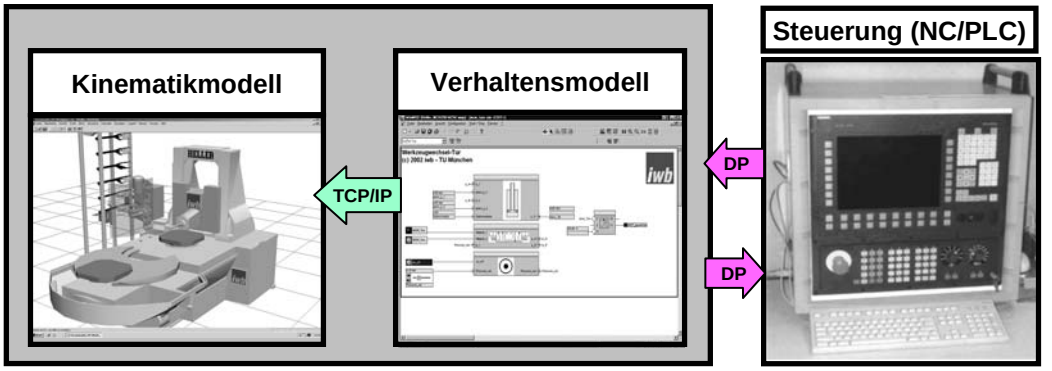
Abbildung 3-15: Modellierungstechnik für die Hardware-in-the-Loop-Simulation [ZAEH & POERNBACHER 2005]

Die einzelnen Baugruppen werden als hierarchisierbare *UML-Komponenten* visualisiert. Jede Komponente besitzt Schnittstellen, so genannte *Ports*, anhand derer sie Informationen mit anderen Komponenten austauschen kann.

Die Ports beschreiben, welche Daten eine Komponente braucht bzw. zur Verfügung stellt. Die *Konnektoren* hingegen beschreiben den Informationsfluss.

Besonders hervorzuheben ist die strikte Trennung zwischen dem Modell für die Steuerungssoftware und dem physikalischen Maschinenmodell. Die meisten Ansätze bieten die Möglichkeit, das Maschinenverhalten durch das Hinterlegen von Zeitangaben als Transitionsbedingungen zu spezifizieren. Dies birgt jedoch eine Reihe von Nachteilen. Auf diese Art und Weise wird das Verhalten der Steuerungsfunktionen mit dem Verhalten der hydraulischen, pneumatischen, elektromechanischen und mechanischen Komponenten vermischt. Eine Wiederverwendung des Modells oder von Teilen davon für weitere Projekte ist somit schwierig, da die einzelnen Baugruppen nicht präzise voneinander getrennt werden können. Darüber hinaus wird durch eine Trennung von steuerungsinternem und maschinenspezifischem Verhalten die reale Schnittstelle an der Maschine im Modell bereits vorweggenommen. Dadurch ist es beispielsweise möglich, bereits im Rahmen der Konzeption Symbollisten, E/A-Listen und Sensor/Aktor-Listen zu generieren [ZAEH & POERNBACHER 2005]. Für die Abbildung der Steuerungsfunktionen werden erweiterte Zustandsgraphen der UML verwendet. Die Abbildung des physikalischen Maschinenverhaltens erfolgt unter anderem mit Zeitgliedern aus der blockorientierten Modellierung und mit so genannten *Look-Up*-Tabellen. Darin werden Eingangswerte direkt entsprechenden Ausgangswerten zugeordnet. Das Modell für die Steuerungssoftware und das physikalische Maschinenmodell sind über definierte Schnittstellen miteinander verbunden. Die Visualisierung der Maschinenabläufe in einem 3D-Modell basiert auf einer Schnittstelle zum Maschinenmodell.

Das vorgestellte Modellkonzept bildet die Grundlage für ein Hardware-in-the-Loop-System für die virtuelle Inbetriebnahme von Werkzeugmaschinen. In Abbildung 3-16 ist der Aufbau der Simulationsumgebung dargestellt.



Kinematiksimulation

Geometrie
(Form, Abmessungen etc.)
Kinematik
(Art, Länge und Anzahl der Achsen, Verfahrbereichsgrenzen etc.)

Verhaltenssimulation

Physikalisches Verhalten
(Zeitverhalten, Schaltverhalten, Bewegungsverhalten etc.)
Kommunikation mit NC/PLC
Buskonfiguration, E/A-Simulation

Steuerung

Vollständige NC/PLC-Funktionalität
Lagesollwerte

Abbildung 3-16: Hardware-in-the-Loop-System zur Validierung und Optimierung von Steuerungsprogrammen [EHRENSTRASSER ET AL. 2003]

Für die Umsetzung wurden kommerzielle Softwarewerkzeuge eingesetzt. Die Erstellung des Kinematikmodells sowie die Visualisierung des 3D-Modells erfolgten mit dem Produkt *eM-Workplace* aus dem Hause Technomatix [TECHNOMATIX 2005]. Unter anderem lassen sich damit serielle Kinematiken mit rotatorischen und translatorischen Achsen anwenderspezifisch definieren. Über eine Programmierschnittstelle können von externen Client-Anwendungen Serverfunktionen des 3D-Kinematik-Simulationssystems aufgerufen werden.

Für die Verhaltenssimulation wird das zuvor erläuterte physikalische Maschinenmodell verwendet und mit Hilfe des Simulationswerkzeugs *WinMOD* aus dem Hause Mewes & Partner abgebildet [MEWES UND PARTNER 2005]. Das Produkt stellt ein Quasi-Echtzeitsimulationssystem für die Automatisierungstechnik auf Basis der Windows-Plattform dar. Durch den Anschluss einer realen Steuerung an das virtuelle Maschinenmodell ist es somit bereits in frühen Phasen der Entwicklung möglich, die korrekte Funktionsweise der entwickelten Steuerungsprogramme zu testen und in nachfolgenden Iterationsschleifen zu verfeinern. Detaillierte Informationen über den Aufbau und die Funktionsweise sind in [REINHART ET AL. 2002B], [EHRENSTRASSER ET AL. 2003] sowie [ZÄH ET AL. 2003] zu finden.

In der Arbeit von [TOMASZUNAS 1998] wird das steuerungstechnische Verhalten von Produktionsmaschinen im Rechner mittels der *Real-Time Object-Oriented Modeling-Methode* (ROOM) [SELIC ET AL. 1994] abgebildet. Das Konzept sieht vor, dass das Modell der Maschine auf einem Rechner, der über eine E/A-Schnittstelle mit einer realen Steuerung verbunden ist, in Echtzeit simuliert wird. Eine Erweiterung dieses Konzeptes hinsichtlich Wiederverwendbarkeit wird in [LINGXIANG 2003] vorgestellt. Hier wird vor allem das Ziel verfolgt, durch eine Systematisierung des Modellierungsprozesses die simulationsgestützte Qualitätssicherung weiter zu vereinfachen und in der Praxis zu verankern. Im Gegensatz zu [ZAEH & POERNBACHER 2005] wird hier das physikalische Maschinenverhalten jedoch nicht von der Beschreibung der Steuerungsfunktionen getrennt.

3.5.4 Zusammenfassung

Moderne ECAD-Systeme bieten erste Ansätze für eine modellgetriebene Vorgehensweise in der Elektrokonstruktion. Die Grundlage hierfür ist eine Datenbank, welche die einzelnen Planarten sowie die Symbolbibliothek verwaltet. Makro-Sprachen erlauben eine teilweise Automatisierung der Tätigkeiten und bieten umfangreiche Möglichkeiten, die Funktion der ECAD-Systeme zu erweitern. Trotz internationaler Standards auf der Grundlage der ISO 10303-212 basieren die Datenschnittstellen weiterhin auf proprietären Lösungen bzw. so genannten Quasi-Standards einzelner Hersteller. Den in Bezug auf die modellgetriebene Entwicklung fortgeschrittensten Ansatz in diesem Bereich stellt die Föderale Informations-Architektur (FIA) des Forschungsprojektes Föderal dar. Allerdings liegt der Fokus hier ausschließlich auf der Elektrokonstruktion. Andere Bereiche, wie die Entwicklung der Steuerungssoftware, werden nicht im selben Maße modelltechnisch unterstützt. Hierfür präsentiert die Arbeit von [ZAEH & POERNBACHER 2005] ein innovatives Modellierungskonzept, das für die Entwicklung von Maschinen- und Steuerungsfunktionen verwendet wird und als Grundlage für eine Hardware-in-the-Loop-Simulation dient. In der Mechanikkonstruktion haben sich Modelle zur Beschreibung bzw. Optimierung von Maschineneigenschaften mittlerweile etabliert. Die Forschungsansätze arbeiten an Konzepten und Lösungen, um deren Durchgängigkeit in der Mechanikkonstruktion weiter zu verbessern und sie um Lösungen aus der Regelungstechnik sowie der Simulation des thermoelastischen Verhaltens zu erweitern.

3.6 Produktdaten-Management (PDM)

Bei der Entwicklung von Werkzeugmaschinen entsteht eine große Menge auf komplexe Weise miteinander in Verbindung stehender Dokumente unterschiedlicher Art [RUMPE 1996]. Die Entwicklungsdokumente geben eine bestimmte Sicht auf die Werkzeugmaschine wieder und bilden spezifische Maschineneigenschaften anhand abstrakter, idealisierter Beschreibungsmittel ab. Bei der Entwicklung von Werkzeugmaschinen sind folglich unterschiedliche *Dokumenttypen* erforderlich, die sich vor allem in den zu Grunde liegenden Strukturierungs- und Darstellungskonzepten unterscheiden. Die einzelnen Fachbereiche verwenden jene Dokumenttypen, mit denen sie den für sie relevanten Ausschnitt der Werkzeugmaschineneigenschaften entwickeln und beschreiben können. Durch die mechatronische Produktstruktur der Werkzeugmaschine stehen die Dokumente jedoch nicht für sich allein, sondern beschreiben technische Details, die sich in mehreren Dokumenten in unterschiedlichen Sichtweisen wiederfinden. In Abbildung 3-17 sind einige Dokumenttypen, aufgeteilt auf die drei Bereiche Mechanikkonstruktion, Elektrokonstruktion und Softwareentwicklung, dargestellt.

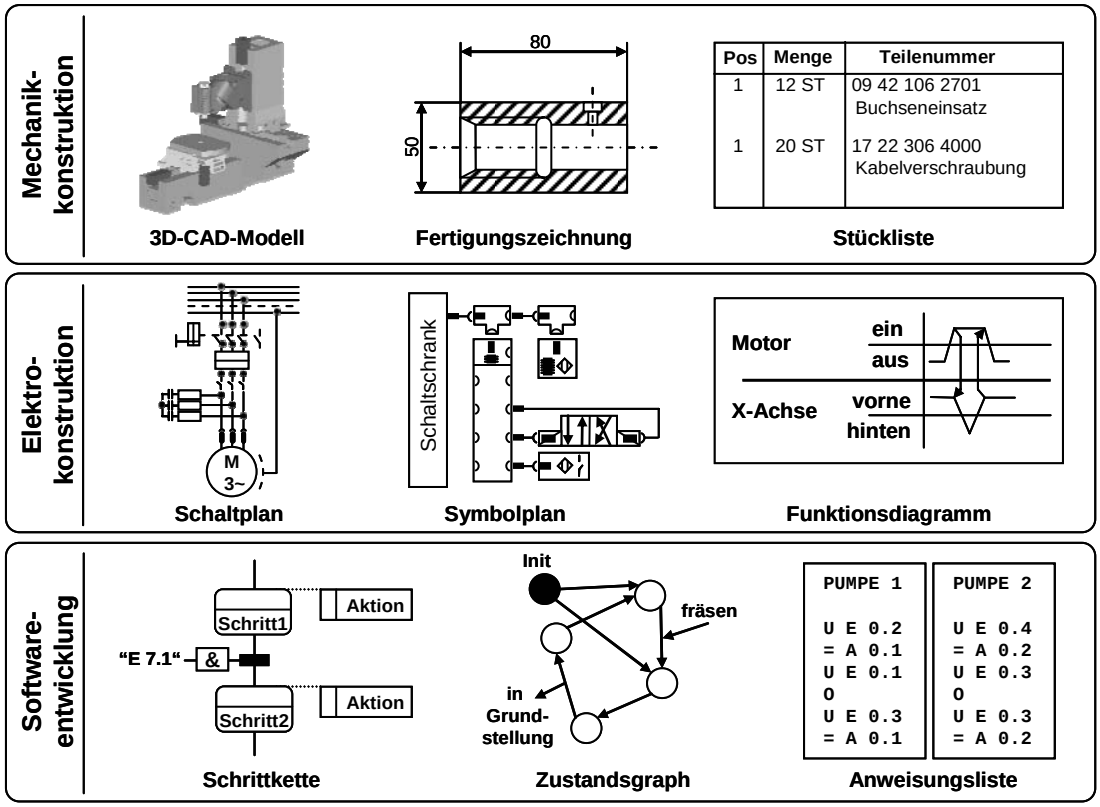


Abbildung 3-17: Unterschiedliche Dokumenttypen in der Werkzeugmaschinenentwicklung

Das 3D-CAD-Modell der Mechanikkonstruktion ist eines der wichtigsten Dokumente in der Werkzeugmaschinenentwicklung und beschreibt das zentrale Konzept der Maschine. Der Konstrukteur legt direkt oder indirekt die Aufstellfläche, den Bauraum, die Anzahl und die Lage der benötigten Achsen sowie die Bewegungsabläufe fest. Im 3D-CAD-Modell werden die wesentliche Produktstruktur und Produktgeometrie abgebildet. Von den hier getroffenen Vorgaben leiten sich die Handlungsfelder der weiteren Fachbereiche ab. So ergibt sich aus der Funktionsstruktur der Maschine die Beschaltung der Motoren bzw. der Antriebe. Je nachdem, ob sich ein Förderband für den Spänetransport in beide Richtungen bewegen soll oder nicht, ändert sich dementsprechend der Schaltplan in der Elektrokonstruktion. Ebenso bestimmen der Bauraum und die Bewegungsabläufe die Festlegung der elektrischen Betriebsmittel. Die Anzahl und der Einbauort der Sensoren beispielsweise hängen davon ab, welche Maschinenzustände erfasst und überwacht werden sollen. Im Installationsplan werden die elektrischen Betriebsmittel sowie die notwendige Infrastruktur, wie beispielsweise E/A-Module, in einer Ein-Liniendarstellung abgebildet und miteinander verbunden. Anhand der Funktionsdiagramme beschreibt die Elektrokonstruktion die Funktionsfolgen und die steuerungstechnische Verknüpfung der elektrischen Betriebsmittel in zwei Koordinaten. In der senkrechten Achse wird der Zustand der Aktoren, der Sensoren und der Stellglieder aufgetragen. Die waagrechte Koordinate entspricht der Zeit des Steuerungsablaufs. Diesem Funktionsdiagramm entnimmt die Softwareentwicklung die Informationen über Abläufe, Maschinenzustände und Zeitvorgaben zur Programmierung der Steuerungssoftware. Die Implementierung selbst erfolgt meist durch eine Sprache der IEC 61131-3, wie beispielsweise die *Anweisungsliste* (AWL), oder durch graphische Beschreibungsmittel, wie die *Schrittkette* (SFC) oder den *Zustandsgraph*.

Die Abhängigkeiten der Entwicklungsdokumenttypen spiegeln die enge technische Verzahnung der Komponenten auf Maschinenebene wider. Auf der organisatorischen Ebene setzt dies effiziente Methoden und Softwarewerkzeuge voraus, um eine konsistente, widerspruchsfreie und transparente Verwaltung der Entwicklungsdokumente gewährleisten zu können. Aus dieser Ausgangssituation heraus gewinnt das Produktdatenmanagement (PDM) zunehmend an Bedeutung.

Mit dem Begriff PDM wird allgemein die Verwaltung, Organisation und Steuerung produktdefinierender Daten (Produktdaten) verstanden, die im Laufe des Produktentwicklungsprozesses entstehen. Das Ziel ist es, eindeutige und reproduzierbare Produktkonfigurationen zu erzeugen [EIGNER & STELZER 2001]. PDM-Systeme verwalten die mit den fachbereichsspezifischen Softwarewerkzeugen erstellten Produktdaten und werden zur Unterstützung der verschiedenen Prozessketten in der Produktentstehung eingesetzt. Im folgenden Abschnitt werden PDM-Systeme und diesbezüglich weitergehende Konzepte vorgestellt.

3.6.1 Produktdaten-Management-Systeme

Nach [SCHÖTTNER 1999] liegt die zentrale Aufgabe von PDM-Systemen darin, die einzelnen CAx-Informationsinseln zu verknüpfen und die Produktentwicklung mit vollkommen rechnergesteuerten Prozessen zu realisieren, durch die Schritt für Schritt ein virtuelles Produktmodell entsteht. Sie dienen als Integrationsplattform von verschiedenen, im Entwicklungsprozess eines Produktes eingesetzten Anwendungssystemen und ermöglichen eine konsistente Datenhaltung [ANDERL ET AL. 2002]. PDM-Systeme stellen hierbei sicher, dass die zur Entwicklung des Produktes notwendigen Informationen während des gesamten Produktlebenszyklus am richtigen Ort, zur richtigen Zeit und in bedarfsgerechter Qualität und Quantität bereitgestellt werden [EIGNER & STELZER 2001].

In der Literatur wird häufig von EDM/PDM-Systemen gesprochen. EDM steht hierbei für *Engineering Data Management* und unterstreicht den prozessorientierten Schwerpunkt bei der Produktdatenverwaltung, während mit PDM die Datenorientierung betont wird. Da in der Praxis eine exakte Trennung dieser beiden Bereiche nicht sinnvoll ist, werden häufig beide Bezeichnungen gemeinsam zur Kennzeichnung von Produktdaten-Management-Systemen verwendet [SCHÖTTNER 1999]. Des Weiteren wird die Abkürzung EDM vereinzelt mit *Engineering Document Management* gleichgesetzt. Diese Systeme sind schwerpunktmäßig für die Verwaltung von digitalisierten Papierdokumenten konzipiert, wobei keine Zuordnung zu Produktstrukturen wie in PDM-Systemen erfolgt [EIGNER & STELZER 2001]. In dieser Arbeit wird für die Bezeichnung von Softwaresystemen, die Produktdaten verwalten, ausschließlich der Begriff *PDM-System* verwendet.

In [VOGEL 2000] werden PDM-Systeme allgemein in dokumentenorientierte und *CAD-nahe* Systeme eingeteilt. Die Entwicklung von PDM-Systemen war ursprünglich aus dem Mechanik- bzw. Konstruktionsbereich motiviert, um die umfangreichen Datenbestände an Zeichnungen und 3D-Modellen verwalten zu können. Dementsprechend ist hier auch die größte funktionale Integration zwischen dem Produktdatenmanagement und den Anwendungssystemen gegeben. CAD-nahe PDM-Systeme werden auch als *Team-Data-Management-Systeme* (TDM-Systeme) bezeichnet.

Architektur

Allgemein lässt sich die Architektur von PDM-Systemen in zwei Ebenen unterteilen. Auf der so genannten Makroebene befindet sich das Produktmodell. Es definiert die Beziehungen zwischen Metadaten und wird deshalb auch als Metamodell des PDM-Systems bezeichnet. Auf der Mikroebene werden die Anwendungsdaten in Form physikalischer Dateien gespeichert. Das Metamodell bildet die zu beschreibenden Daten auf die Anwendungsdaten ab und verweist auf die physikalischen Dateien, welche die jeweiligen Dokumente repräsentieren. Als Metadaten werden unter anderem der Ersteller eines Dokuments, das Erstellungsdatum, der Freigabestatus sowie der Pfad zur physikalischen Datei gespeichert [ANDERL ET AL. 2002], [KRASTEL 2002], [KLEINER 2003].

Funktionsüberblick

Die zentralen Aufgabenbereiche von PDM-Systemen sind in [SVENSSON 2003], [AHLE 2000] und [KRASTEL 2002] aufgeführt. Insgesamt lassen sich folgende Punkte zusammenfassen:

- Dokumentenmanagement
- Struktur- und Konfigurationsmanagement
- Datenmanagement (Metadatenmanagement)
- Varianten- und Versionsmanagement
- Projekt-, Workflow- und Prozessmanagement
- Datenbankmanagement (Datensicherung, Datenschutz)
- Visualisierungsmanagement
- Schnittstellenmanagement

Das Dokumentenmanagement ist eine der grundlegenden Funktionen eines PDM-Systems. Es ordnet die verschiedenen Entwicklungsdokumente in die Produktstruktur ein. Die Dokumente selbst werden dabei als Container betrachtet, welche die eigentlichen Daten und Informationen kapseln [BLUDAU & WELP 2005]. Das Dokument ist gewissermaßen eine Black-Box, deren Inhalt extern anhand von Metadaten beschrieben wird. Für die Verwaltung von CAD-Dokumenten sind weitergehende Funktionen verfügbar, die beispielsweise das Auslesen der Produktstruktur in Datensätze erlauben, die in ihrer Repräsentation am Rechner als Baugruppen und Bauteile visualisiert werden. Eine bedeutende Rolle im Dokumenten- sowie im Struktur- und Konfigurationsmanagement spielt das Nummernsystem. Nach [DIN 6580 1985] dient eine Nummer zur eindeutigen Identifikation eines Objekts, beispielsweise durch eine Sach- und Artikelnummer für Einzelteile oder Baugruppen. Darüber hinaus werden Nummernsysteme auch zum Aufbau einer funktions- und ortsbezogenen Kennzeichnungssystematik verwendet [DIN 6779 1992]. Sie erlauben eine eindeutige Bezeichnung von Baugruppen und Bauteilen nach Aufgabe, Funktion, Art und Einsatzort festzulegen. Ein Kennzeichen ist hierbei eine eindeutige Nummer, die in einen oder mehrere Kennzeichnungsblöcke unterglie-

dert ist. Ein Kennzeichnungsblock ist eine gegliederte Zusammenfassung zusammengehöriger Informationen. In Abbildung 3-18 ist der Aufbau eines Kennzeichnungsblocks zu sehen.

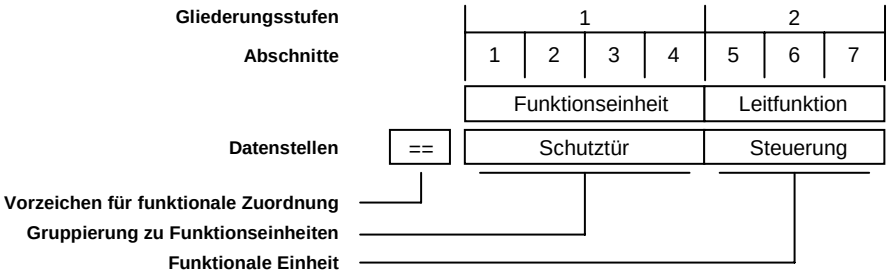


Abbildung 3-18: Kennzeichnungsblock „Funktionale Zuordnung“ nach [DIN 6779 1992]

Der Kennzeichnungsblock *Funktionale Zuordnung* dient zur Kennzeichnung von funktionalen Zusammenhängen zwischen Anlagen und technischen Einrichtungen (z. B. elektrische Betriebsmittel) und wird mit dem Vorzeichen “= =“ dargestellt. Mit dem Vorzeichen “++“ wird eine ortsbezogene und mit “--“ eine produktbezogene Kennzeichnung formuliert. Eine vollständige Übersicht ist in [DIN 6779 1992] enthalten. Gliederungsstufen ordnen die in den Abschnitten aufgeführten Funktionseinheiten und Funktionen bestimmten Systemen, Subsystemen oder technischen Einrichtungen zu.

Ein weiterer wesentlicher Bestandteil eines PDM-Systems ist das *Konfigurationsmanagement*. Eine Konfiguration beschreibt dabei den Status eines Produkts zu einem beliebigen Zeitpunkt. In [DIN ISO 10007 2004] wird das Konfigurationsmanagement als eine Managementdisziplin beschrieben, die über die gesamte Lebensdauer eines Produktes angewandt wird, um die Transparenz und die Überwachung seiner funktionellen und physischen Merkmale sicherzustellen. Das Konfigurationsmanagementmodul implementiert hierzu Funktionen, welche die zu einer Produktkonfiguration gehörenden Informationen zusammenfassen und verwalten. Dies setzt ein effizientes Datenmanagement der einzelnen Dokumente und der sie charakterisierenden Metadaten voraus. Letztere werden unter anderem für das Änderungsmanagement eingesetzt, das dafür sorgt, dass Änderungen identisch am Produkt und den dazugehörigen Dokumenten durchgeführt werden. Die Vorgehensweise bei der Änderung von Dokumenten bzw. Objekten ist in den verschiedenen PDM-Systemen ähnlich. Der Begriff *Änderung* ist hier stets im Sinne von Versionierung zu verstehen [EIGNER & STELZER 2001]. Das Varianten- und Versionsmanagement dient dazu, geänderte bzw. ältere Versionen eines Dokuments oder Objekts im PDM-System zu speichern und verfügbar zu halten. Die Freigabe- und Ein- und Auscheckmechanismen werden von einem Workflowmanagementsystem unterstützt. Ein Workflow beschreibt einen Geschäftsprozess, der aus einzelnen Aktivitäten besteht. Diesen sind verschiedene ausführende Ressourcen (Personen, Rollen), zu benutzende Ressourcen (Dokumente) und Tätigkeiten zugeordnet. Das Workflowmanagementsystem steuert den Ablauf von Workflow-Prozessen, indem es den einzelnen Ressourcen die vordefinierten Aktivitäten zur Bearbeitung vorlegt. Die Speicherung der Geschäftsprozesse im PDM-System führt somit zu einer ISO 9000-gerechten Dokumentation der Einführungs- und Änderungsvorgänge [AHLE 2000].

Von besonderer Bedeutung für das PDM-System ist dessen Fähigkeit, Daten mit zahlreichen fachspezifischen Softwarewerkzeugen auszutauschen. Die Voraussetzung dafür ist eine Vielzahl von Schnittstellen, die im Rahmen eines geeigneten Integrationskonzepts implementiert werden müssen.

Integrationskonzepte

Das PDM-System bildet das zentrale Informationssystem in einem Unternehmen. Aus diesem Grund kann dessen Einführung und Anwendung nicht singulär betrachtet werden, sondern muss fest in der IT-Strategie und den Entwicklungsabläufen des Unternehmens verankert sein. Insbesondere die Anbindung fachbereichsspezifischer Softwarewerkzeuge an das PDM-System spielt in diesem Zusammenhang eine wesentliche Rolle. In [EIGNER & STELZER 2001] sind die drei wesentlichen Formen der Anbindung von PDM- an Anwendungssysteme aufgeführt. Abbildung 3-19 zeigt deren prinzipielle Funktionsweise.

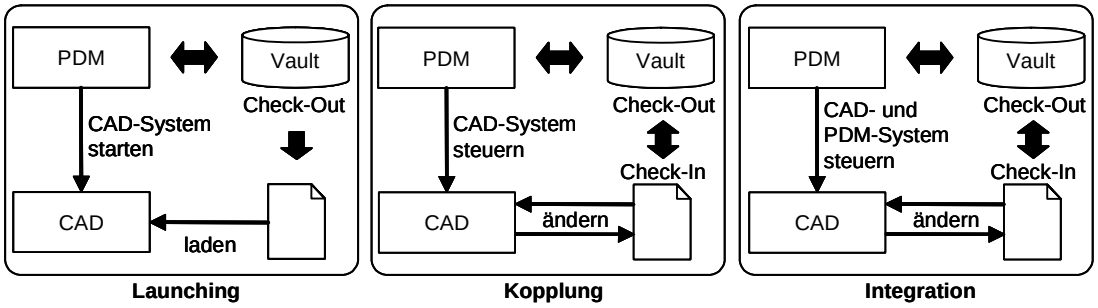


Abbildung 3-19: Formen des Zugriffs auf externe Softwarewerkzeuge [EIGNER & STELZER 2001]

Die einfachste Form der Anbindung stellt das *Launching* dar. Der Anwender nutzt das PDM-System dabei, um das Dokument zu finden, es auszuchecken und in das dazugehörige Softwarewerkzeug zu laden. Der Datenaustausch erfolgt dabei durch den Im- und Export von Dokumenten. Das PDM-System muss hierzu über geeignete Konverter bzw. Schnittstellen verfügen, um die verschiedenen Datenformate der Dokumente verarbeiten zu können.

Bei der *Kopplung* wird die API der Softwarewerkzeuge verwendet, um Zugriff auf die internen Informationen der jeweiligen Daten zu erlangen. Im PDM-System ist es dadurch möglich, die Daten auszuwerten und z. B. die Produktstruktur aus den CAD-Modellen aufzubauen. Kopplungen erlauben auch die Rückspeicherung (Check-In) in das PDM-System.

Die *Integration* ist die höchste Form der Verbindung zweier Systeme. Der Unterschied zur Kopplung liegt hierbei in der bidirektionalen Kommunikation zwischen dem PDM-System und einem fachspezifischen Softwarewerkzeug. So ist es beispielsweise möglich, dass der Anwender Funktionen des PDM-Systems in der Benutzeroberfläche des CAD-Systems aufrufen kann, etwa den Produktstruktur-Browser [EIGNER & STELZER 2001]. Die hierfür notwendige technische Infrastruktur basiert auf der API der eingesetzten Softwarewerkzeuge bzw. auf kommunikationsorientierter Middleware wie beispielsweise CORBA. Diese direkte Form der Anbindung findet sich vorwiegend im CAD-Bereich.

Zur Integration der Daten aus den kaufmännischen und logistischem Prozessketten (beispielsweise Kosten, Lagerbestand oder Hersteller) bieten die PDM-Systeme Schnittstellen zu *Enterprise-Resource-Planning*-Systemen (ERP-Systemen). Andererseits dringen genau diese Anbieter betriebswirtschaftlicher Standardsoftware zunehmend in den Bereich der Produktdatenverwaltung mit entsprechenden PDM-Funktionen vor. Die datentechnische Integration mit dem jeweiligen ERP-System ist bei diesen Systemen direkt gegeben, da das PDM-System als ein Modul des ERP-Systems betrachtet wird [KRASTEL 2002]

In den Arbeiten von [KLEINER 2003] und [GERHARD 2000] wird zu den drei vorgestellten Anbindungsformen noch das Konzept der Föderation autonomer Produktmodelle bzw. PDM-Systeme hinzugefügt. Die produktbeschreibenden Daten werden von den einzelnen Softwarewerkzeugen angelegt und liegen somit verteilt und lokal in den jeweiligen Fachbereichen vor. Die Verwaltung der Daten erfolgt anhand der unterschiedlichen Produktmodelle der verschiedenen Softwarewerkzeuge. Allgemein kann von einer föderierten Integration gesprochen werden, wenn einzelne lokale IT-Systeme innerhalb einer übergeordneten Instanz zugreifbar gemacht werden, aber dennoch eigenständig und autonom sind. Der Datenaustausch erfolgt durch Import- und Exportschemata, wobei die Kooperation zwischen den verschiedenen Softwarewerkzeugen durch einen Föderationsdienst realisiert wird [GERHARD 2000]. Dieser Ansatz basiert auf den Konzepten föderierter Datenbanken und dem daraus entstandenen föderativen Produktdatenmanagement [MONTAU 1996].

Neben der datentechnischen Integration verfügen PDM-Systeme noch über aufgaben- und datenflussorientierte Prozess- und Projektmanagementkomponenten. Diese so genannten Workflowmanagement-Systeme integrieren die einzelnen Entwicklungsabläufe auf der prozesstechnischen Ebene und erlauben die Planung und Steuerung der einzelnen Softwarewerkzeuge im Prozess- und Arbeitsablauf [KLEINER 2003].

Erweiterte Anwendungsbereiche und Konzepte

Durch die zunehmende Globalisierung und Dynamisierung der Geschäftsbereiche steigen die Anforderungen an PDM-Systeme. Insbesondere die interkulturelle Zusammenarbeit über Unternehmens- und Ländergrenzen hinweg sowie die geforderte Zugriffsmöglichkeit auf verteilte Produktdaten in verschiedenen PDM-Systemen führen dazu, dass bestehende Konzepte durch neue Technologien und Standards erweitert werden müssen.

In [GRABOWSKI ET AL. 2003] werden die speziellen Probleme der interkulturellen Zusammenarbeit aufgeführt. Neben sprachlichen Barrieren und unterschiedlichen Ausbildungsstandards werden vor allem voneinander abweichende Vorgehensweisen bei der Problemlösung, Unwissenheit über die lokale Gesetzgebung, unterschiedliche Firmenstandards und Regelungen sowie divergierende Begriffsbezeichnungen angeführt. Internationale Kooperationen zeichnen sich durch eine besondere Dynamik in den Geschäftsprozessen aus, die von den PDM-Systemen nicht erfasst werden können, zumal deren Einsatz ein gewisses Maß an *Datenqualität* (Konsistenz und Vollständigkeit) voraussetzt. Dies kann auf Grund des unterschiedlichen Entwicklungswissens der Mitarbeiter und der voneinander abweichenden Geschäftsprozesse nicht gewährleistet werden. Darüber hinaus erschweren heterogene IT-Infrastrukturen den effizienten und bedarfsgerechten Austausch von Daten und Informationen. Die Produktdaten werden häufig verteilt in den PDM-Systemen der an der Kooperation beteiligten Unterneh-

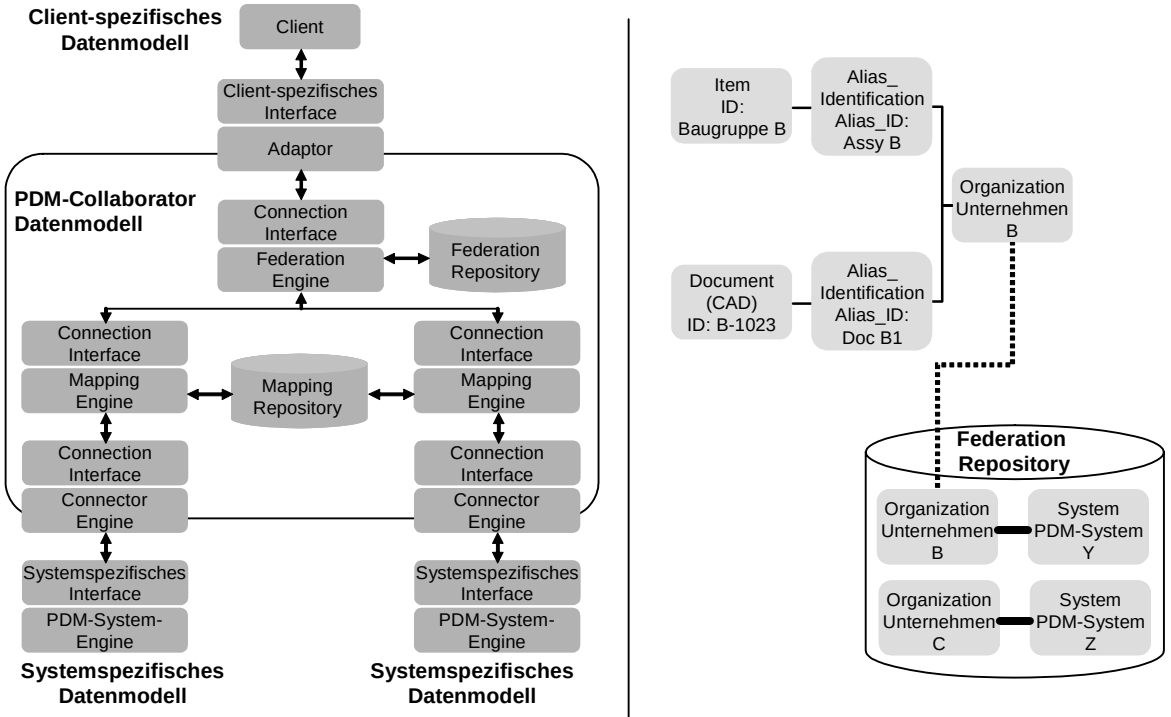
men gespeichert. Damit das Datenmanagement nachvollziehbar und beherrschbar bleibt, müssen Lösungen für den Produktdatenaustausch zwischen den verschiedenen PDM-Systemen sowie ein einheitlicher Zugriffsmechanismus erarbeitet werden.

Der Datenaustausch auf der Grundlage eines einheitlichen Produktmodells wird in [FELDVOSS 2003] vorgestellt. Bei der Entwicklung des Großraumflugzeuges A380 von Airbus Industries wurden verteilt auf die verschiedenen Standorte mehrere PDM- und TDM-Systeme eingesetzt, welche die Daten unterschiedlicher CAD-Systeme verwalten. Einmal im Monat wurden die einzelnen PDM- und TDM-Systeme „manuell“ synchronisiert und die Daten auf einen gemeinsamen Stand gebracht. Für die automatisierte Synchronisierung aller Airbus-Partner wurde ein gemeinsames Datenmodell auf der Grundlage der ISO 10303-AP214 erarbeitet, das die Produktstruktur, entsprechende Metadaten sowie die Daten zur Visualisierung und für das Konfigurationsmanagement beschreibt. Somit war es möglich, die einzelnen Entwicklungsdaten täglich abzugleichen und die Entwicklungsabläufe effizienter zu gestalten.

Nach [GRABOWSKI ET AL. 2003] reicht der Datenaustausch über standardisierte Formate allein nicht aus, da internationale Normen wie die ISO 10303 die Semantik bestimmter Datenelemente nicht präzise vorschreiben. Durch unternehmensspezifische Regeln oder Vereinbarungen (wie beispielsweise Attributwerte für Nummernsysteme zu belegen sind) oder durch die Definition zusätzlicher unternehmensspezifischer Attribute kann die Interpretation der Daten variieren [KRAUSE ET AL. 2003]. In [EHRLER ET AL. 1999] wird die Problematik analysiert und Methoden und Werkzeuge zur Lösung werden vorgeschlagen. Diese sind jedoch mit einem großen Aufwand verbunden und somit nur für langfristige Kooperationen zwischen den Geschäftspartnern sinnvoll.

Für die Handhabung verteilter Produktdaten wird in [NOWACKI & VON LUKAS 2003] und [KRAUSE ET AL. 2003] der Einsatz föderierter PDM-Systeme für den Aufbau eines virtuellen PDM-Systems vorgeschlagen. Dieses speichert selbst keine Daten, sondern verteilt die Datenanfragen an die einzelnen PDM- bzw. CAx-Systeme und fügt die partiellen Suchergebnisse zu einem einheitlichen Suchergebnis zusammen. Voraussetzung hierfür ist die Abbildung der einzelnen, voneinander abweichenden Datenmodelle der PDM-Systeme in einem gemeinsamen, uniformierten bzw. föderierten Modell. Ebenso müssen die verschiedenen Schnittstellen für den Datenzugriff vereinheitlicht werden. Die Grundlagen für die Lösung dieser Probleme wurden in dem Projekt *PDTnet* [PDTNET 2005] erarbeitet. Hier wurde als wesentliches Ergebnis eine einheitliche PDM-Schnittstelle entwickelt, die auf einem neutralen Datenmodell nach STEP AP 214 für PDM-Produktdaten basiert. Ein Web-Client erlaubt den parallelen Zugriff auf mehrere PDM-Systeme sowie eine einheitliche Sicht auf die Daten. Ein vergleichbares Konzept wird in [KRAUSE ET AL. 2003] vorgestellt. Im Rahmen des Verbundprojektes *PDM-Collaborator* wird ein Lösungsansatz präsentiert, der die Kommunikation kooperierender Unternehmen auf der Basis einer verteilten, heterogenen PDM-Umgebung ermöglicht. Prinzipiell wird hier zwischen einer produktdatengetriebenen, einer projektdatengetriebenen und einer regelbasierten Föderation unterschieden. Die Einteilung erfolgt danach, wo und wie die Regeln zur Verteilung der Anfragen auf die verschiedenen PDM-Systeme erfasst werden. Bei der produktdatengetriebenen Föderation muss die Information, die in den Produktdaten der einzelnen Systeme vorhanden ist, ausgewertet werden, um feststellen zu können, ob zu einem bestimmten Objekt der Produktstruktur in anderen PDM-Systemen weitere

Informationen vorhanden sind. Diese Objekte bilden somit die Schnittstelle zwischen den einzelnen Teil-Produktstrukturen und liegen redundant in den verschiedenen PDM-Systemen vor. Die Beziehungen zwischen diesen Teil-Produktstrukturen werden durch Elemente des Datenmodells STEP AP 214 formuliert, da das PDM-Collaborator-Datenmodell (siehe Abbildung 3-20 a) eine Untermenge davon ist. Mit dem Element *Alias_Identification* können Referenzen auf die verteilten Teil-Produktstrukturen formuliert werden. Es erlaubt die Identifikation von Datenobjekten in verschiedenen Kontexten, beispielsweise in einem anderen Unternehmen. Abbildung 3-20 b zeigt die Informationen, die bei der Föderation mit diesem Element notwendig sind.



a) PDM-Collaborator-Datenmodell b) Für die Föderation notwendige Daten

Abbildung 3-20: Produktdatengetriebene Föderation mit dem PDM-Collaborator-Datenmodell [KRAUSE ET AL. 2003]

Für die eindeutige Identifikation der Daten in einer verteilten PDM-Umgebung sind Informationen

- zur Identifikation des gesuchten Datenobjekts im anderen System,
- zum Unternehmen, in dem weitere Produktdaten verwaltende PDM-System verwendet werden,
- und zum PDM-System selbst

erforderlich. In den Produktdaten werden normalerweise keine Informationen hinterlegt, die beschreiben, in welchen PDM-Systemen weitere Produktdaten zu finden sind. Diese müssen zusätzlich zu den Produktstrukturen in einem *Federation Repository* gespeichert werden. Vor allem für die unternehmensübergreifende Kooperation erweist sich der föderative Ansatz als

vorteilhaft. Die einzelnen CAx-Systeme bleiben autonom und können fachbereichsspezifisch angepasst werden [GERHARD 2000].

Die Anwendung von PDM-Systemen darf jedoch nicht nur auf konstruktionsnahe Prozesse beschränkt bleiben. Gerade durch die Bedeutung der Mechatronik und die zunehmende Komplexität der Software im Produktentwicklungsprozess sind neue bzw. erweiterte Datenverwaltungskonzepte erforderlich [ANDERL ET AL. 2002]. Die Herausforderung liegt darin, die Software in ein einheitliches Konzept zur Produktdatenverwaltung einzubetten und somit als zentralen Bestandteil der Produktstruktur zu betrachten. In den Arbeiten von [SVENSSON & CRNKOVIC 2002] und [SIELAFF 2003] werden Lösungskonzepte vorgestellt, wie die Softwareentwicklung in die Mechanikentwicklung eingebunden werden kann. Beiden Konzepten liegt die Idee zu Grunde, PDM-Systeme mit *Software-Configuration-Management*-Systemen (SCM) zu verknüpfen. Ein Vergleich der Funktionalität dieser beiden Systeme ist in [DAHLKVIST 2001] zu finden. SCM-Systeme werden für die Verwaltung von Source-Code eingesetzt und stellen unter anderem Funktionen für die Versionierung, das Release-Management und das verteilte Arbeiten verschiedener Entwicklergruppen zur Verfügung. Das zentrale Element zur Integration von PDM- und SCM-Systemen stellt in [SVENSSON & CRNKOVIC 2002] die Systemstruktur dar, die vom PDM-System verwaltet wird. Die einzelnen *Parts* und *Assemblies* (siehe Anhang 11.3.3) lassen sich einfach darauf abbilden. Vom SCM-System werden darüber hinaus die Source-Code-Klassen, ausführbare Programme sowie der Status des Lebenszyklus dieser Elemente referenziert. Die Klassen können im PDM-System wie Parts behandelt und auf dieselbe Art und Weise gespeichert werden. Der Status im Lebenszyklus wird anhand von Metadaten als Attribut eingefügt. Die ausführbaren Programme werden auf einem Fileserver gespeichert und vom PDM-System referenziert. Dieses Konzept wurde in [SIELAFF 2003] an einem konkreten, objektorientierten PDM-System umgesetzt. Hierzu wurde das Datenmodell des PDM-Systems um die Strukturen der zu verwaltenden Programmiersprache erweitert und die zentralen Sprachelemente auf die PDM-Elemente abgebildet. Ein Programm entspricht demnach einem Assembly, eine Klasse einem Part und die Abhängigkeiten zwischen den Klassen werden durch die Produktstrukturen abgebildet. Dadurch, dass einzelne Elemente der Programmiersprachen auf einzelne PDM-Elemente abgebildet werden, ist eine sehr feingranulare Verwaltung von Source-Code möglich. Das Abbilden von Softwarearchitekturen oder die Darstellung von Softwareverhalten anhand spezifischer Modellierungstechniken ist jedoch nur indirekt gegeben.

Eine Weiterentwicklung im PDM-Bereich wurde im Rahmen des vom BMBF geförderten Forschungsprojekts *Innovative Technologien und Systeme für die integrierte virtuelle Produktentwicklung* (iViP) durchgeführt. Ziel des Projektes war es, eine durchgängige Entwicklungsplattform für alle Phasen der Produktentstehung zu erarbeiten [KRAUSE ET AL. 1998]. Ein Schwerpunkt lag im Management aller anfallenden Produktdaten und der Integration fachspezifischer Softwarewerkzeuge. In Abbildung 3-21 ist das wesentliche Konzept dargestellt. Das Management der Produktdaten wird von PDM-Basisdiensten übernommen. Diese gliedern sich in zwei Bereiche. Im PDM-Kern sind die grundlegenden Funktionen eines PDM-Systems implementiert. Mit den Definitionswerkzeugen kann das PDM-System an betriebliche Randbedingungen angepasst werden [EHRLER ET AL. 1999].

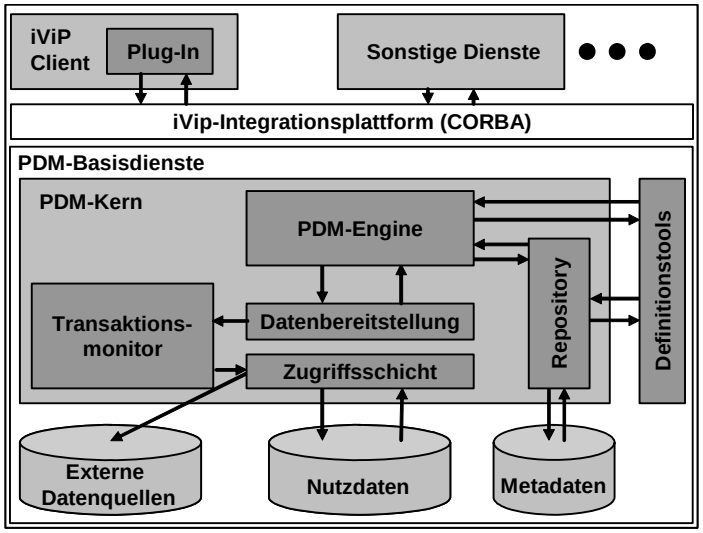


Abbildung 3-21: Grundkonzeption der iViP-Integration [EHRLER ET AL. 1999]

Das Konzept der iViP-Plattform geht von einer sehr engen Integration der Kooperationspartner sowie deren IT-Infrastruktur aus. Flexible Kooperationen aus einzelnen KMUs stehen nicht im Fokus [GERHARD 2000]. In den folgenden vier Abschnitten werden nun die Schnittstellen vorgestellt, die beim Einsatz eines PDM-Systems von Bedeutung sind.

3.6.2 STEP-PDM-Schema

Für den Austausch von CAD-Daten zwischen verschiedenen CAD-Systemen wird von den Unternehmen häufig das neutrale STEP-Format eingesetzt. Vor allem die Automobilindustrie verwendet STEP, um komplexe Baugruppenstrukturen in verschiedenen Softwarewerkzeugen zu verarbeiten. Der Austausch der Daten zwischen unterschiedlichen PDM-Systemen über Unternehmensgrenzen hinweg ist problematischer, da die Datenmodelle der PDM-Systeme an die Anforderungen der jeweiligen Unternehmen angepasst sind und somit semantische Unterschiede aufweisen können. Der Bedarf, administrative PDM-Daten weiterzugeben, steigt jedoch, da durch die internationalen Geschäftsbeziehungen der Unternehmen die Wertschöpfungskette größer und komplexer wird. Daten, die von den diversen PDM-Systemen eingelesen und interpretiert werden können, erleichtern die Steuerung der weltweiten Geschäftsprozesse. Um eine Standardisierung in diesem Bereich zu forcieren, wurde die Entwicklung des STEP-PDM-Schemas vorangetrieben. Dieses ist ein Referenz-Informationsmodell für den Austausch einer allgemeinen bzw. zentralen Untermenge von Daten, die innerhalb von PDM-Systemen verwaltet wird [GERHARD 2000]. Das Ziel ist es, die Interoperabilität zwischen verschiedenen STEP-Applikationsprotokollen zu verbessern und zu harmonisieren [KINDRICK ET AL. 2000]. PDM-relevante Daten wie Materialdaten, Produktstrukturen oder Klassifizierungen sind fachbereichsübergreifend und werden dadurch in mehreren Anwendungsprotokollen beschrieben. Beim Datenaustausch kann dies zu Fehlern und Inkonsistenzen führen. Das STEP-PDM-Schema stellt daher eine Untermenge der Anwendungsprotokolle 214, 212, 203 und 232 dar (siehe Abbildung 3-22).

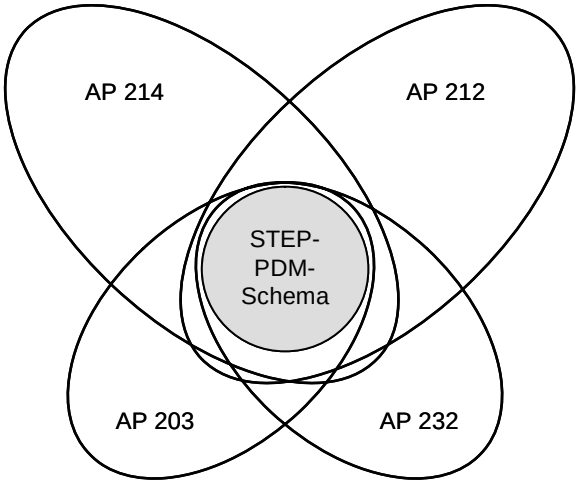


Abbildung 3-22: Umfang des STEP-PDM-Schemas

Das STEP-PDM-Schema legt als Modell die wesentlichen Entitäten und Attribute fest, die für den PDM-Datenaustausch benötigt werden. Dies sind insbesondere Eigenschaften, Strukturen und Beziehungen von Objekten sowie deren Identifikation, Versionierung und Klassifikation. Darüber hinaus werden Informationen zur Konfiguration und zur Effektivität von Strukturen sowie zur Authentisierung und Organisation von Personen, Gruppen, Rollen, Projekten usw. festgelegt. Der Datenaustausch basiert auf der in der ISO 10303-21 (siehe Anhang 11.3.3) definierten Syntax.

3.6.3 OMG PDM-Enablers

Die PDM-Enablers-Spezifikation [OMG 2000] der *Object Management Group* (OMG) legt einen Standard für die Funktionalitäten von PDM-Systemen fest. Das Ziel ist es, verschiedenen Anwendungssystemen den Zugriff auf Funktionen eines PDM-Systems zu ermöglichen sowie den Aufbau und die Implementierung verteilter PDM-Systeme auf Basis der *Common Object Request Broker Architecture* (CORBA) zu unterstützen [KRASTEL 2002]. PDM-Enabler bieten demnach einen standardisierten Zugang zu den Diensten von PDM-Systemen für die verschiedenen CAx-Anwendungen in einem Unternehmen. Sie definieren hierzu ein Modell von PDM-Schnittstellen in der programmiersprachenunabhängigen *Interface Definition Language* (IDL), welche auf kommerziell erhältliche PDM-Systeme abgebildet werden können.

- Die Schnittstellendefinitionen stellen unter anderem folgende Dienste bereit:
- Check-In- und Check-Out-Mechanismen sowie Sperrfunktionen für Produktdokumente wie CAD-Modelle und Anforderungsspezifikationen,
- Persistenz und Archivierung von Produktstrukturen bzw. einzelner Bauteile und deren Beziehungen sowie
- Versionierung von Produktstrukturen und Änderungsmanagement.

Bauteile bzw. Baugruppen werden im PDM-Enabler-Ansatz durch ihre Struktur, durch Attribute, wie z. B. IDs oder Bezeichnungen, und durch zugeordnete Dokumente beschrieben. Ihre innere Struktur wird nicht verwaltet und bleibt anderen Standards wie STEP oder XML vorbehalten [GERHARD 2000].

Die PDM-Enablers-Spezifikation ist modular aufgebaut und umfasst insgesamt zwölf Teilmodelle, die als *Enabler* bezeichnet werden. Das Gesamtmodell wird dabei in drei Basismodule und neun Domainmodule unterteilt, die von den Basismodulen durch Vererbung abgeleitet werden. In Abbildung 3-23 ist der prinzipielle Aufbau der PDM-Enablers-Spezifikation zu sehen.

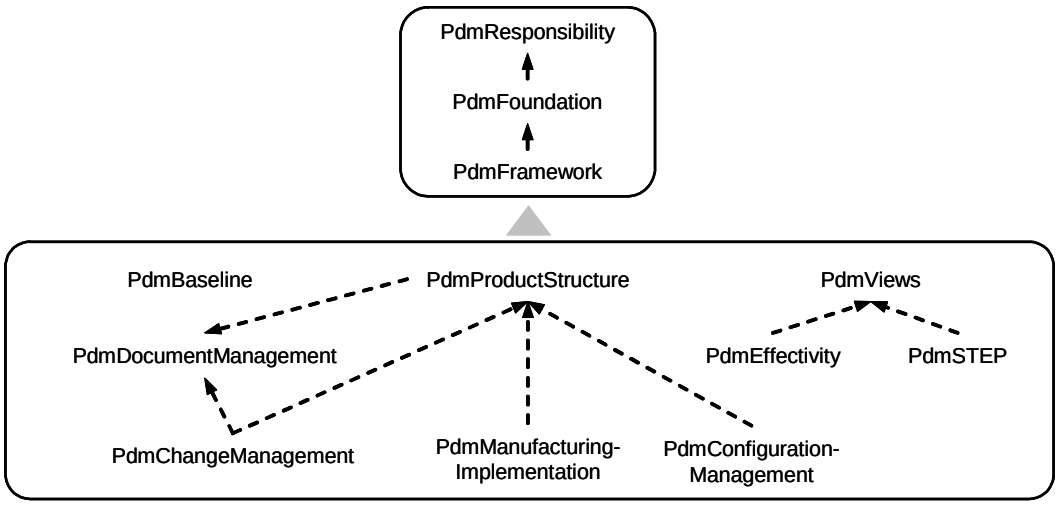


Abbildung 3-23: Module der PDM-Enabler [OMG 2000]

Jedes Modul wird in einem eigenen UML-Paket mit entsprechenden Klassendiagrammen beschrieben. Das Basismodul *PdmResponsibility* definiert eine minimale Schnittstelle, die von den Enablern zur Referenzierung von Rollen und Organisationen als Besitzer, Ansprechpartner oder weiteren Verantwortlichkeiten verwendet wird. Die Klassen im *PdmFoundation*-Modul spezifizieren elementare Verhaltensbeschreibungen, die von zahlreichen PDM-Enablern benötigt werden. Dabei wurden verschiedene Typen von Verhaltensbeschreibungen identifiziert und in eigenen Klassen abgebildet. Die Klasse *Manageable* spielt hierbei eine zentrale Rolle. Sie spezifiziert eine minimale Verhaltensbeschreibung, die ein PDM-Objekt der Instanzebene aufweisen muss, um mit Hilfe der PDM-Enabler manipuliert zu werden. Im Modul *PdmFramework* werden PDM-spezifische Frameworkobjekte und Relationsklassen spezifiziert. Hier wird auch das Master-Revision-Iteration-Konzept zur Verwaltung von Teilen (Items) und Dokumenten beschrieben [GERHARD 2000]. Die *ItemMaster-Klasse* beschreibt Objekte, die unabhängig von Revisionen oder möglichen Änderungen vom PDM-System verwaltet werden. Sie enthält Attribute, die sich über den gesamten Lebenszyklus des Elements nicht ändern, z. B. Identifikatoren oder Betriebsmittelkennzahlen. Die Klasse *ItemRevision* kapselt hingegen Attribute, die sich beispielsweise bei Freigabe- oder Änderungsprozessen ändern. Jedes der ItemRevision-Objekte ist mit einem einzelnen, korrespondierenden ItemMaster verknüpft. Iterationen werden in der *ItemIteration-Klasse* repräsentiert. Iterationen entstehen bei jedem schreibenden Zugriff auf ein Objekt und dienen dazu, den Lebenszyklus eines Elements zu dokumentieren. Die Domainmodule sind direkt von den Basismodu-

len abgeleitet und verwenden deren Funktionalität zur Implementierung spezifischer Dienste. Eine genaue Beschreibung der einzelnen Module ist in [OMG 2000] zu finden.

Der Fokus der PDM-Enablers-Spezifikation liegt in der Interoperabilität zwischen mehreren, parallel laufenden Applikationen. Im Gegensatz zum STEP-PDM-Schema wird der Datenaustausch über Dateien nicht abgedeckt. Des Weiteren werden durch die IDL-Schnittstellen Operationen auf den PDM-Objekten definiert. Sie bilden somit eine funktionale Sicht auf die Produktdaten ab. Das STEP-PDM-Schema hingegen legt ausschließlich das Datenmodell der PDM-Objekte und deren Beziehungen fest. Die Applikationsprotokolle der ISO 10303 und das PDM-Enabler Konzept sind jedoch noch nicht durchgängig aufeinander abgestimmt. So deckt keines der existierenden Applikationsprotokolle die PDM-Enablers-Spezifikation in der erforderlichen Breite ab. Da diese wiederum viel Spielraum bei der Implementierung zulassen, kann es zu semantischen Inkonsistenzen zwischen den Systemen kommen. Die ISO 10303-22 bietet mit dem *Standard Data Access Interface* (SDAI) ebenfalls die Möglichkeit, online auf Produktdaten zugreifen zu können. Allerdings wird diese Schnittstelle von den PDM-Anbietern derzeit nicht unterstützt [GERHARD 2000].

3.6.4 MechaSTEP

Das Projekt *STEP Datenmodelle zur Simulation mechatronischer Systeme* (MechaSTEP) wurde 1997 von der Automobilindustrie initiiert. Sein Ziel war es, ein fachbereichsübergreifendes Datenmodell auf der Grundlage der ISO 10303 zu entwickeln, um Simulations- und Berechnungsdaten aus den Bereichen Mechanik, Elektrotechnik, Hydraulik und Regelungstechnik abzubilden. Die Kernanforderungen bei der Konzeption von MechaSTEP waren die präzise Definition interdisziplinärer Schnittstellen in den Modellbeschreibungen sowie die Darstellung der fachbereichsübergreifenden Verzahnung mechatronischer Komponenten. MechaSTEP definiert in diesem Zusammenhang ein Sichtenkonzept, das es erlaubt, die einzelnen Arbeitsschwerpunkte der Fachbereiche zu beschreiben und disziplinbezogen darzustellen. Wesentlich ist, dass MechaSTEP eine vollständig parametrisierbare Abbildung der Modellbeschreibungen unterschiedlicher Fachbereiche unterstützt. Verhaltensbeschreibungen von mechatronischen Komponenten werden in MechaSTEP durch mathematische Gleichungen fachbereichsübergreifend abgebildet.

Um das Datenmodell von MechaSTEP als Metamodell für ein PDM-System nutzen zu können, sind einige Erweiterungen notwendig. Unter anderem weist MechaSTEP Defizite in der Verwaltung von Konfigurationen auf. Des Weiteren besteht keine Möglichkeit der Versionierung von Modellelementen sowie der Experiment- und Ergebnisverwaltung [KRASTEL 2002].

3.6.5 Product Data Markup Language (PDML)

Die *Product Data Markup Language* (PDML) ist eine XML-Sprache, die eine Sprachdefinition zum Austausch organisatorischer und strukturbezogener Produktdaten zwischen verschiedenen PDM-Systemen ermöglicht. Sie wurde 1999 unter der Federführung des *US Department of Defense* (DoD) im Rahmen des Projekts *Product Data Interoperability* (PDI) entwickelt. In der PDML werden sieben so genannte *Application Transaction Sets* (ATS) definiert.

Das sind spezifische DTDs (siehe Anhang 11.3.1), welche die Datenstrukturen dedizierter Softwarewerkzeuge beschreiben. Des Weiteren umfasst die PDML ein Integrationsschema in EXPRESS und Transformationsregeln, um die ATS auf das Integrationsschema abbilden zu können. Das Integrationsschema ist ebenfalls eine DTD und basiert auf den *Integrated Resources* der ISO 10303. Die PDML stellt ein Standard-Vokabular zur Verfügung, um PDM-Daten über das Internet austauschen zu können. Prinzipiell ist dies auch mit einer STEP-Datei nach ISO 10303-21 möglich. Der Vorteil der PDML liegt in der einfacheren Verarbeitung der XML-Strukturen durch freie Standardsoftwarewerkzeuge, einer einfacheren Lesbarkeit und der Möglichkeit, Informationen über das zu Grunde liegende Schema in der Datei selbst zu hinterlegen. Der Nachteil der PDML liegt in dem fest vorgegebenen Schema [GERHARD 2000].

3.6.6 Product Life Cycle Management (PLM)

Die in den Unternehmen eingesetzten Integrationslösungen kommen vorwiegend aus den Bereichen PDM, *Enterprise Resource Planning* (ERP), *Supply Chain Management* (SCM) und *Customer Relationship Management* (CRM). Die Grenzen zwischen diesen Systemen sind nicht immer eindeutig zu ziehen. Es wird versucht, alle Bereiche von der Konstruktion und Produktion über den Vertrieb bis zur Instandhaltung und der Außerbetriebnahme unter dem Begriff *Product Lifecycle Management* (PLM) zusammenzuführen. In [ARNOLD ET AL. 2005] wird PLM als ein integrierendes Konzept zur IT-gestützten Organisation aller Informationen über Produkte und deren Entstehungsprozesse über den gesamten Produktlebenszyklus hinweg verstanden. Das Ziel ist es, Informationen immer aktuell an den relevanten Stellen im Unternehmen zur Verfügung stellen zu können.

PLM ist ein Gesamtkonzept, das in die Wertschöpfungskette der Unternehmen integriert werden muss. Es integriert einzelne Systeme wie PDM und ERP zu einer Gesamtlösung für das Informationsmanagement. Charakteristisch ist eine produktzentrierte Sicht auf alle produktbeschreibenden Daten und Informationen unter Berücksichtigung der informationsverarbeitenden Prozesse über den gesamten Lebenszyklus hinweg. In Abbildung 3-24 ist dieser Sachverhalt graphisch dargestellt.

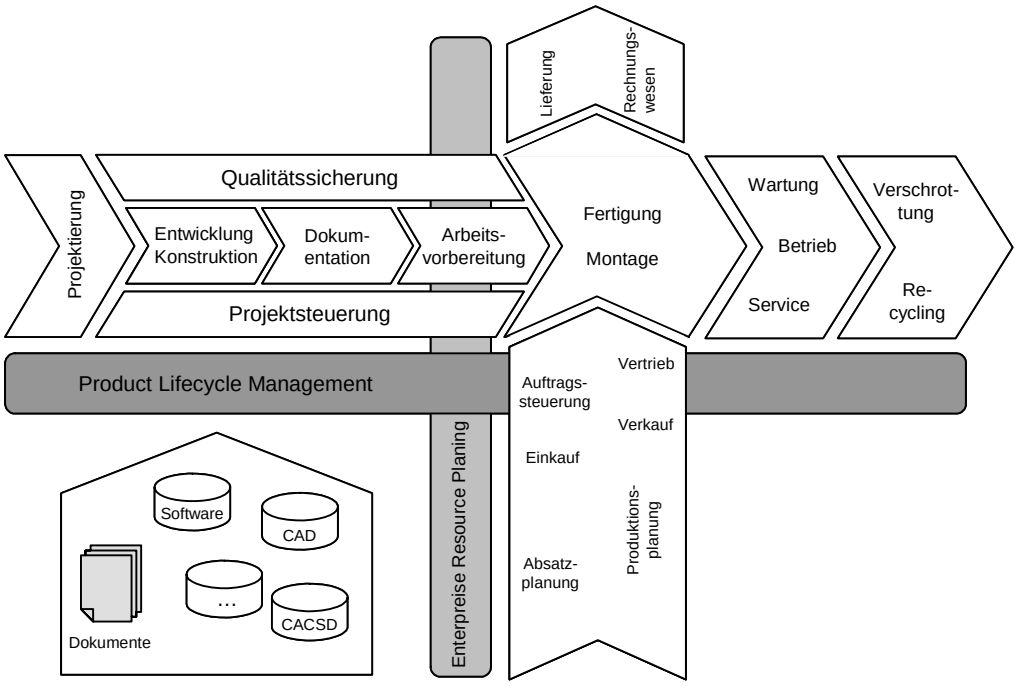


Abbildung 3-24: Konzept des Product Lifecyle Managements [ARNOLD ET AL. 2005]

In den Unternehmen werden häufig zwei Kernprozesse herausgestellt, die sich in bestimmten Bereichen überschneiden. Das PLM fokussiert dabei die Produkte mit ihren Entstehungsprozessen, während das ERP den Bereich der Produktion adressiert. Die Schnittstelle zwischen diesen beiden Kernprozessen wird dem PLM zugerechnet [ARNOLD ET AL. 2005].

3.6.7 Zusammenfassung

Die im Entwicklungsprozess anfallenden Dokumente werden vor allem in der Automobil- und Luftfahrtindustrie mit PDM-Systemen verwaltet. Insbesondere im Zuge des Product Lifecycle Managements stellt das PDM-System den zentralen Datenspeicher dar. Die Anbindung unterschiedlicher Softwarewerkzeuge aus den verschiedenen Fachbereichen ist daher ein bedeutender Erfolgs- und Effizienzfaktor, um die Abstimmungsprozesse verteilter Produktentwicklungsprozesse zu optimieren. Historisch bedingt bieten PDM-Systeme zahlreiche Funktionen, um die Entwicklungsdaten der Mechanikkonstruktion durchgängig zu verwalten. So können einzelne Baugruppen und Bauteile in einer Produktstruktur zusammengefasst und mit den entsprechenden CAD-Modellen referenziert werden. Darüber hinaus werden zunehmend auch Daten aus der Elektrokonstruktion integriert, um beispielsweise die Planung umfangreicher Kabelbäume aus der Automobil- und Luftfahrtindustrie in die Entwicklungsabläufe einzubinden. Integrationslösungen für den Bereich der Softwareentwicklung haben sich jedoch noch nicht in der erforderlichen Breite durchsetzen können. Für den Austausch von PDM-Daten sind keine proprietären Formate vorhanden, so dass hier die entsprechenden Applikationsprotokolle der ISO 10303 eingesetzt werden. Da diese jedoch sehr komplex sind, stellen nur wenige PDM-Systeme entsprechende Schnittstellen zur Verfügung, wodurch deren Bedeutung in der Praxis sehr gering ist.

3.7 Ableitung des Handlungsbedarfs

In den in Abschnitt 3.4 aufgeführten Arbeiten ist die *Funktion* der zentrale Begriff der Modellbildung. Die Funktion wird als das integrative Element für den interdisziplinären Abstimmungsprozess definiert und stellt die Schnittstelle der fach- und disziplinspezifischen Aufgabenfelder dar. Zahlreiche Arbeiten nehmen diesen Umstand als Grundlage, um Vorgehensweisen und Modellierungstechniken zur Beschreibung mechatronischer Prinziplösungen oder zur Entwicklung von Steuerungssoftware zu erarbeiten. Von der Anforderungs- bis zur Konzeptphase sowie in der Detaillierung und Konstruktion können durchgängig Modelle zur Beschreibung und Abbildung von Maschineneigenschaften sowie zur Maschinenkonfiguration und -simulation eingesetzt werden. Insbesondere nutzen die vorgestellten Arbeiten formale bzw. semiformale Beschreibungsmittel, um die erarbeiteten Modellierungstechniken und -sprachen dokumentieren zu können. Dadurch ist eine präzise und korrekte Interpretation der Modelle möglich.

Neben geeigneten Softwarewerkzeugen, welche die Modelle erstellen können, sind entsprechende Integrationslösungen notwendig, welche die fachspezifischen Modelle ineinander überführen können. Auf diese Weise wird die Weiterverarbeitung einzelner Modellbeschreibungen in anderen Fachbereichen ermöglicht. In Abschnitt 3.6 wurden verschiedene Konzepte und Lösungsansätze für die Integration von Produktdaten und Modellbeschreibungen aufgeführt. Sie basieren zum Großteil auf kommerziell erhältlichen PDM-Systemen verschiedener Hersteller und bauen auf dem internationalen Standard ISO 10303 auf. Auf Grund der in Kapitel 3 herausgearbeiteten Schwachstellen eignen sich diese Ansätze nur bedingt, um eine durchgängige modellgetriebene Entwicklung von Werkzeugmaschinen zu etablieren. Insbesondere eine fachbereichsübergreifende semantische Konsistenz in Bezug auf die eingesetzten Modellbeschreibungen ist mit den aktuellen Integrationslösungen nur in Teilbereichen möglich. Im Rahmen dieser Arbeit sollen deshalb Methoden und Werkzeuge erarbeitet werden, die den Aufbau und den Einsatz einer Entwicklungsplattform für die modellgetriebene Entwicklung von Werkzeugmaschinen ermöglichen. Im Wesentlichen ergeben sich folgende Handlungsfelder:

- Definition einer Vorgehensweise und eines Systems zur formalisierten Beschreibung der in den einzelnen Fachbereichen eingesetzten Modellierungssprachen (*Domain Specific Languages*)
- Erarbeiten eines Konzepts und eines Systems zur Transformation der fachspezifischen Modellbeschreibungen in jeweils andere Formate
- Konzeption und Implementierung eines Repositories zur zentralen Datenhaltung
- Zusammenführen der erarbeiteten Lösungskonzepte für die Datenintegration in einen Konzeptvorschlag für den Aufbau einer Entwicklungsumgebung zur modellgetriebenen Entwicklung von Werkzeugmaschinen

Ausgehend von diesen Handlungsfeldern wird in den folgenden Kapiteln das Konzept einer integrierten Entwicklungsplattform als Grundlage für die modellgetriebene Entwicklung von Werkzeugmaschinen vorgestellt.

4 Die virtuelle Werkzeugmaschine

4.1 Übersicht

Zu Beginn dieses Kapitels wird der in dieser Arbeit entwickelte Begriff der *virtuellen Werkzeugmaschine* definiert und als Plattform für die modellgetriebene Entwicklung von Werkzeugmaschinen vorgestellt. Im Anschluss daran werden die Anforderungen an die Datenintegrations-Komponente der virtuellen Werkzeugmaschine formuliert, bevor eine formale Herleitung ihrer Funktionsweise und ihres Aufbaus erfolgt. Abschließend wird das Konzept der vorgeschlagenen Integrationslösung zusammenfassend dargestellt.

4.2 Einführung

Bisher existiert keine allgemein gültige Definition des Begriffs „virtuelle Werkzeugmaschine“. Entsprechend des fachlichen Hintergrunds der einzelnen Verfasser werden unterschiedliche Aspekte und Sichtweisen hervorgehoben. In [GROßMANN 2002] wurden die bislang dazu veröffentlichten Aussagen kritisch behandelt, jedoch selbst wiederum nur der Versuch einer Definition unternommen. Im Folgenden ist die am *iwb* erarbeitete Definition ausgeführt. Diese gründet auf den langjährigen Forschungsaktivitäten bezüglich der verschiedenen Aspekte bei der Entwicklung von Werkzeugmaschinen. Hervorzuheben sind im Wesentlichen die Schwerpunkte der Modellbildung und Simulation des Bewegungs- und Strukturverhaltens, der Umsetzung von Maschinenabläufen [ZÄH ET AL. 2003], der Modellierung von Steuerungssoftware sowie des methodischen Vorgehens im Rahmen der Entwicklung [REINHART ET AL. 1998], [REINHART ET AL. 1999].

Definition 4-1: Virtuelle Werkzeugmaschine [ENGLBERGER ET AL. 2003]:

> Unter der virtuellen Werkzeugmaschine wird ein Informationsmanagementsystem zur ganzheitlichen Entwicklung von Werkzeugmaschinen verstanden. Die virtuelle Werkzeugmaschine hat die Aufgabe, eine durchgängige Modellierung, Berechnung, Simulation und Visualisierung der Baugruppen, deren Funktionsweise und des Verhaltens der realen Gesamtmaschine zu ermöglichen.
>
> Der Aufbau besteht aus einem zentralen Datenmodell und den damit verknüpften Modellierungs-, Berechnungs-, Simulations- und Visualisierungswerkzeugen. Auf Basis des zentralen Datenmodells erfolgt das koordinierte Zusammenspiel der in diesem Aufbau enthaltenen Rechnerwerkzeuge. Dieses koordinierte Zusammenwirken erweitert das Systemverhalten der virtuellen Werkzeugmaschine gegenüber der isolierten Anwendung der einzelnen Werkzeuge.

Im Sinne der Definition ist die virtuelle Werkzeugmaschine eine rechnergestützte Entwicklungsumgebung, welche die Arbeiten und Ergebnisse der an der Entwicklung einer Werkzeugmaschine beteiligten Fachbereiche systematisch in einem gemeinsamen Kontext integriert (siehe Abbildung 4-1). Ziel ist der durchgängige Einsatz von der ersten Produktidee bis zur fertigen Maschine. Die virtuelle Werkzeugmaschine unterstützt die Produktentwicklung in

den Phasen der Konzeption, des Entwurfs und der Ausarbeitung und geht über eine reine Simulationsanwendung hinaus. Die für mehrere Fachbereiche relevanten Informationen, die in den Entwicklungsphasen von den einzelnen Fachbereichen generiert bzw. benötigt werden, werden automatisiert in das gemeinsame Modell integriert bzw. dort abgerufen.

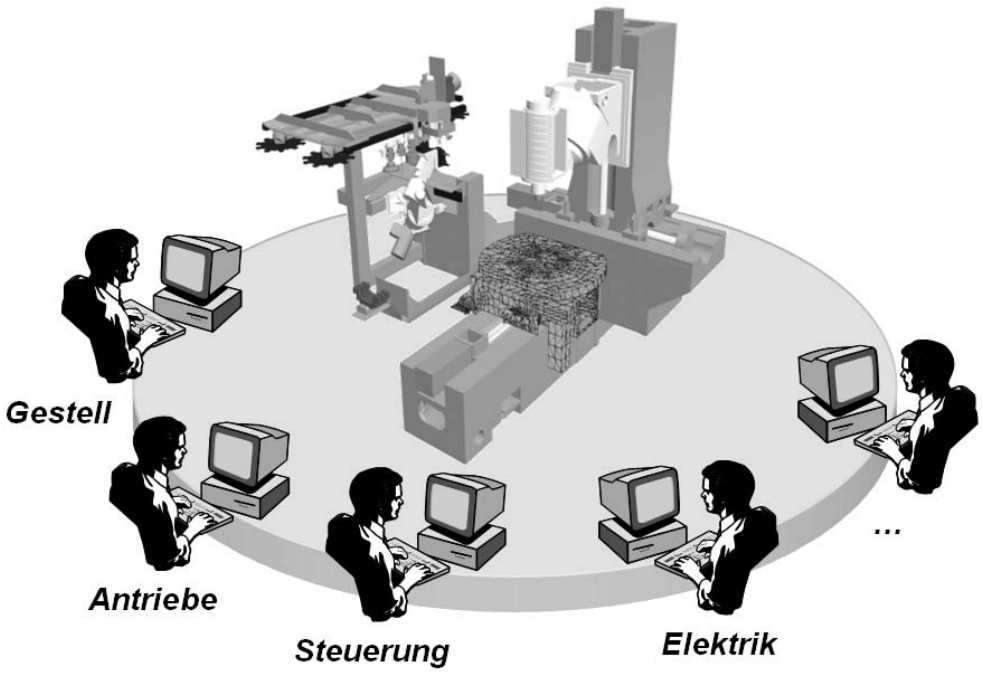


Abbildung 4-1: Die virtuelle Werkzeugmaschine als integrierter Entwicklungsarbeitsplatz ([ENGLBERGER ET AL. 2003])

Als relevante Informationen werden diejenigen Informationen betrachtet, die der Abstimmung und Freigabe der Entwicklungsarbeiten zwischen den Fachbereichen dienen. Die Interaktion des Anwenders mit der virtuellen Werkzeugmaschine und damit die Datengenerierung sowie der Datenabruf erfolgen über die fachspezifischen Rechnerwerkzeuge (siehe Abbildung 4-1). Damit können die Fachbereiche jeweils mit ihrer gewohnten Arbeitsumgebung interagieren. Das bisherige Entwicklungsvorgehen innerhalb des Fachbereichs kann somit beibehalten werden, die Modellierung wird aber durch die konsistente und systematische Datenhaltung der Gesamtmaschine wesentlich beschleunigt.

4.3 Aufbau und Anwendung der virtuellen Werkzeugmaschine

Die virtuelle Werkzeugmaschine ist eine Client-Server-Anwendung und besteht insgesamt aus drei logischen Ebenen. Auf der Anwenderebene befinden sich die spezifischen Softwarewerkzeuge der einzelnen Fachbereiche (siehe Abbildung 4-2). Diese sind expliziter Teil der integrierten Entwicklungsumgebung und bilden die Clients, mit denen die verschiedenen Eigenschaften der realen Werkzeugmaschine beschrieben werden. Jedes Softwarewerkzeug stellt eine eigene Sicht auf die Maschine dar und steuert somit einen Ausschnitt zum virtuellen Abbild der realen Werkzeugmaschine bei. Die Kopplung der verschiedenen Clients erfolgt mittels so genannter XML-Adapter. Diese haben die Aufgabe, die unterschiedlichen

Schnittstellen in eine einheitliche Form zu übersetzen, so dass die importierten Daten im zentralen Datenmodell verarbeitet werden können. Dieses befindet sich in der Datenebene der virtuellen Werkzeugmaschine. Der Datenaustausch erfolgt auf der Grundlage von XML-Dokumenten, die von den jeweiligen Clients bzw. von den XML-Adaptern erzeugt und in das zentrale Datenmodell importiert werden. Die XML-Notation ist innerhalb der virtuellen Werkzeugmaschine formal definiert und legt die inneren Strukturen der einzelnen Dokumente offen (White-Box-Sicht).

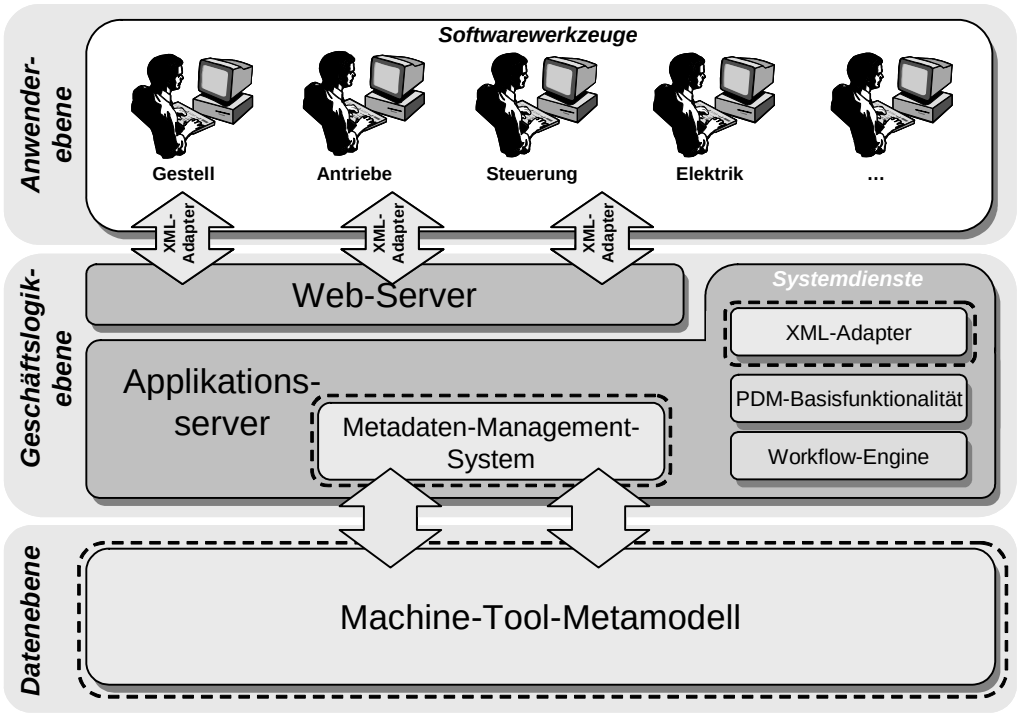


Abbildung 4-2: Aufbau der virtuellen Werkzeugmaschine

Die XML-Adapter sind auf der Geschäftslogikebene als so genannte *Systemdienste* ausgeführt. Systemdienste sind zentrale Funktionen, die der Anwenderebene zur Verfügung gestellt werden. Dazu gehören auch Funktionen für das Projektmanagement, für die Versionsverwaltung sowie für den Ein- und Auscheck-Mechanismus von Dokumenten. Eine Workflow-Engine erlaubt es, unternehmensspezifische Standardabläufe zu definieren und mit den Daten der virtuellen Werkzeugmaschine zu verbinden. Ziel ist ein automatisierter Informationsfluss zur vereinfachten Weitergabe, zur Bearbeitung und zum Abschluss von Arbeitsabläufen innerhalb strategischer, funktionaler oder administrativer Geschäftsprozesse. Die Systemdienste werden von einem Applikationsserver verwaltet. Dieser bietet ein geeignetes Framework für deren Implementierung und ermöglicht den Anwendern zusammen mit einem Web-Server den unternehmensweiten Zugriff auf die Informationen der Datenebene.

Die Anwendung der virtuellen Werkzeugmaschine ist ähnlich der eines PDM-Systems. Der Anwender erstellt mit einem fachbereichsspezifischen Softwarewerkzeug ein Entwicklungsdokument, das in die virtuelle Werkzeugmaschine geladen bzw. eingecheckt wird. Der hierzu notwendige Ablauf ist in Abbildung 4-3 skizziert. Das Einlesen der Entwicklungsdokumente erfolgt anhand der XML-Adapter durch einen geeigneten Parser. Dies setzt voraus, dass sich

die Dokumente durch entsprechende Grammatiken beschreiben lassen, wie zum Beispiel Dokumente in XML-Notation oder nach der internationalen Norm ISO 10303-21 (STEP). Für die interne Weiterverarbeitung der Daten werden die Dokumente in ein neutrales Zwischenformat, die *Virtual Machine Tool Language* (VMTL), transformiert. Dies geschieht automatisch ohne Eingaben von Seiten des Anwenders. Die Sprachelemente der VMTL sind im zentralen Datenmodell definiert. Der Einsatz eines neutralen Zwischenformats bietet den Vorteil, dass die Abhängigkeiten der Daten über Dokumentgrenzen hinweg einfacher und somit effizienter überprüft werden können, da die Datenstrukturen der einzelnen Dokumente bereits aufeinander abgestimmt sind.

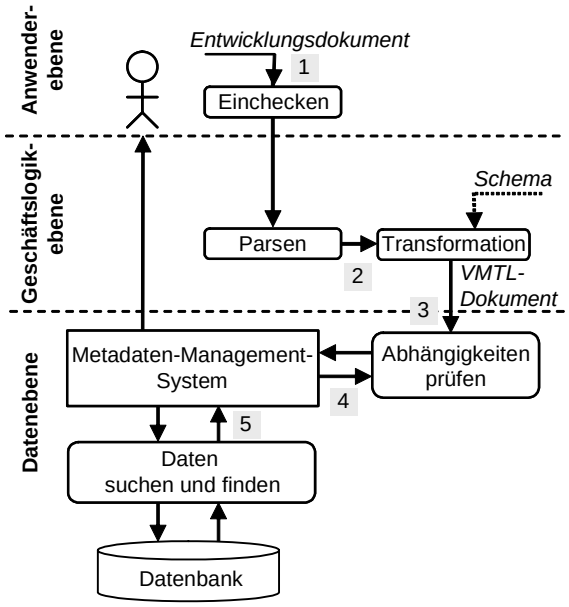


Abbildung 4-3: Funktionsweise der virtuellen Werkzeugmaschine

Die Abhängigkeiten zwischen den Daten sind in einem zentralen Datenmodell abgebildet. Dieses so genannte *Machine Tool Metamodell* (MTM) wird von einem geeigneten Subsystem (Metadaten-Management-System) verwaltet. Die Prüfung auf Vollständigkeit und Widerspruchsfreiheit der Daten kann dadurch erfolgen, dass das Metamodell mit den Daten in den Dokumenten abgeglichen wird. Treten Abhängigkeiten in den Dokumenten auf, die im Metamodell nicht hinterlegt sind, so ist die Konsistenzprüfung fehlgeschlagen. Des Weiteren stellt das MTM die Grundlage dar, um aus einzelnen Fragmenten verschiedener Entwicklungsdokumente ein neues Dokument generieren zu können. Welche Fragmente aus fachlicher Sicht sinnvoll zu einem neuen Dokument zusammengestellt werden können, wird vom Metadaten-Management-System anhand der im Metamodell enthaltenen Abhängigkeitsbeziehungen festgelegt. Dokumente aus einzelnen Fragmenten anderer Dokumente zusammenzustellen ist somit ein wesentlicher Aspekt der Datenintegration innerhalb der virtuellen Werkzeugmaschine. Das Metamodell sorgt hierbei für die semantische Integration der Daten, während die XML-Adapter diese Aufgabe durch die Transformation in die VMTL auf syntaktischer Ebene umsetzen.

Im Rahmen dieser Arbeit wird ein Konzept vorgeschlagen und evaluiert, das die Funktionsweise der virtuellen Werkzeugmaschine auf der Daten- bzw. Geschäftslogikebene festlegt und

die Problematik der Datenintegration behandelt (siehe Abbildung 4-2, mit Strichlinien gekennzeichnete Bereiche). Die Grundlage hierfür stellt das in der Definition nach [ENGLBERGER ET AL. 2003] aufgeführte zentrale Datenmodell dar. Es ermöglicht den Zusammenschluss einzelner Softwarewerkzeuge zu einem Informationsverbund und die Definition von Randbedingungen und Einschränkungen bezüglich der im Datenmodell hinterlegten Entwicklungsdaten. Die Verwaltung des zentralen Datenmodells übernimmt das bereits erwähnte *Metadaten-Management-System*, das in den folgenden Abschnitten vorgestellt wird.

4.4 Anforderungen an das Metadaten-Management-System

Für den Aufbau eines Informationsverbundes im Rahmen der virtuellen Werkzeugmaschine sind zwei wesentliche Aspekte von Bedeutung. Zum einen muss das Problem der unterschiedlichen Begriffswelten der einzelnen Fachbereiche berücksichtigt werden. Dies ist notwendig, da nur durch eine über alle Fachbereiche abgestimmte Semantik die Konsistenz der Daten gewährleistet und ein einheitlicher Zugriff auf diese ermöglicht wird. Zum anderen bedarf es eines darauf aufbauenden technischen Konzepts, das die Integration der Entwicklungsdaten ermöglicht und die einzelnen Softwarewerkzeuge miteinander verknüpft. Dies schließt eine Lösung der Schnittstellenproblematik mit ein, wodurch Daten und Dokumente mit verschiedenen Formaten gelesen und geschrieben werden können. Obwohl die genannten Aspekte eng miteinander verbunden sind, ist zunächst eine gesonderte Betrachtung sinnvoll, um ein differenziertes Problemverständnis herbeizuführen.

Die Problematik unterschiedlicher Begriffswelten in den diversen Fachbereichen lässt sich anhand des semiotischen Dreiecks [OGDEN & RICHARDS 1923] entsprechend Abbildung 4-4 aufzeigen. Dieses illustriert die Interaktion zwischen Worten bzw. Symbolen, Begriffen und realen Dingen in der Welt.

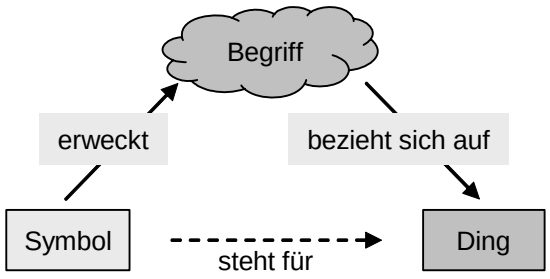


Abbildung 4-4: Das semiotische Dreieck [OGDEN RICHARDS 1923]

Nach [LYONS 1995] ist die Verbindung zwischen einem Wort und einem realen Gegenstand willkürlich durch die Vorstellung der kommunizierenden Personen gegeben. Der Begriff erhält erst durch das Erfahrungswissen der Kommunikationspartner eine konkrete Bedeutung [BÜHLER 1965]. Dieser Sachverhalt trifft ebenso auf die Kommunikationsprozesse bei der Entwicklung von Werkzeugmaschinen zu. Das Erfahrungswissen der Mitarbeiter wird maßgeblich durch ihre Ausbildung bestimmt. Diese entwickeln dementsprechend verschiedene Denkstrukturen und Herangehensweisen an bestimmte Probleme. Bei der Kommunikation zwischen den Mitarbeitern verschiedener Fachbereiche im Rahmen der Abstimmung der Entwicklung kommt es deshalb häufig zu Missverständnissen. Der Begriff der *Komponente*

besitzt beispielsweise für die Mechanikkonstruktion und die Softwareentwicklung unterschiedliche Bedeutung. Der Konstrukteur assoziiert mit diesem Begriff vorwiegend geometrische Eigenschaften, die sich in der Formgebung, in der Gestaltung von Arbeitsräumen oder auch bei der Festlegung von Bewegungsabläufen manifestieren. Die Softwareentwicklung hingegen bezeichnet damit ein "Stück" Software, das über festgelegte Schnittstellen und ein definiertes *Verhalten* auf Umwelteinflüsse reagiert. In Bezug auf das semiotische Dreieck besitzen die beiden Fachbereiche entsprechend ihres Erfahrungswissens unterschiedliche Sichten auf dieselben Dinge. Für den Konstrukteur ist die Komponente *Werkzeugwechsel-Einrichtung* ein mechanischer Funktionsträger, der in einem CAD-Modell beschrieben wird und eine definierte Kinematik zum Wechsel der Werkzeuge besitzt. Der Softwareentwickler hingegen fokussiert in seinen Betrachtungen den Softwareanteil der Komponente, welche die vom Konstrukteur vorgesehenen Bewegungsabläufe automatisiert. Die eingesetzten Softwarewerkzeuge tragen diesem Umstand Rechnung, indem sie sich an der Begriffswelt, den Normen sowie den Modellierungssprachen und Vorgehensweisen der jeweiligen Anwender orientieren. Dieser Umstand stellt einen zentralen Handlungsbedarf bei der datentechnischen Integration der verschiedenen Fachbereiche dar. Das Metadaten-Management-System als Kernkomponente der Integrationslösung der virtuellen Werkzeugmaschine (siehe Abbildung 4-2) muss geeignete Konzepte bieten, um die unterschiedlichen Sichten, Begrifflichkeiten und Modellierungssprachen der Fachbereiche zu integrieren.

Die darauf aufbauende Aufgabe der virtuellen Werkzeugmaschine liegt darin, die Daten der einzelnen Fachbereiche zusammenzuführen. Dies umfasst nicht nur den durchgängigen Datenaustausch zwischen den verschiedenen Softwarewerkzeugen, sondern darüber hinaus die semantische Integration dieser Daten. Damit ist die Fähigkeit gemeint, die einzelnen Entwicklungsdaten in einen fachlichen Zusammenhang zu setzen und ihre Bedeutung für die Spezifikation der Maschineneigenschaften zu erfassen. Das Metadaten-Management-System ist dadurch in der Lage, Widersprüche und Inkonsistenzen in den Modellbeschreibungen fachbereichsübergreifend festzustellen. Voraussetzung hierfür ist eine formale, rechnerinterpretierbare Darstellung der Entwicklungsdokumente in einem zentralen Datenmodell. Eine derartige Beschreibung muss es folglich ermöglichen, die in den Entwicklungsdokumenten enthaltenen Daten in konkrete Modellierungselemente bzw. Strukturierungsmittel einzuordnen. Das System muss in die Lage versetzt werden, zu "verstehen", was in dem Dokument beschrieben wird und welche fachlichen Abhängigkeiten zu weiteren Dokumenten bestehen.

Für den Im- und Export der Dokumente ist weiterhin ein geeignetes Schnittstellenkonzept zu entwerfen, um die Softwarewerkzeuge mit geringem Aufwand in die virtuelle Werkzeugmaschine integrieren zu können. Dabei ist zu berücksichtigen, dass das erarbeitete Konzept auch für ältere Softwaresysteme des Unternehmens funktionieren soll. Wie in Abbildung 4-2 dargestellt, wird der virtuellen Werkzeugmaschine über Systemdienste eine PDM-Basisfunktionalität zur Verfügung gestellt. Das Metadaten-Management-System muss entsprechende Operationen verfügbar machen, auf die diese Systemdienste zugreifen können, um die PDM-Funktionen zu implementieren.

Neben den technischen Anforderungen sind auch wirtschaftliche Aspekte zu berücksichtigen, welche die gesamte virtuelle Werkzeugmaschine und das Metadaten-Management-System im Besonderen betreffen. Da die Einführung einer Integrationslösung in einem Unternehmen

immer mit besonderen Risiken verbunden ist, ist eine schrittweise Umsetzung der entwickelten Konzepte eine wesentliche Voraussetzung. Schwierigkeiten in den Geschäftsprozessen der Entwicklung, bedingt durch die Einführung des Systems, führen zu erheblichen Zusatzkosten, die sich die mittelständische Werkzeugmaschinenindustrie nicht erlauben kann.

4.5 Formale Herleitung des Funktionsprinzips des Metadaten-Management-Systems

Um ein grundlegendes Verständnis über die Funktionsweise der virtuellen Werkzeugmaschine und das Metadaten-Management-System im Besonderen vermitteln zu können, werden in diesem Abschnitt grundlegende formale Konzepte eingeführt, welche die Wechselwirkungen zwischen den Komponenten des Metadaten-Management-Systems beschreiben.

Die wesentlichen Erläuterungen und Definitionen sind den Arbeiten von Yannis Tzitzikas entnommen. In [TZITZIKAS ET AL. 2001], [TZITZIKAS 2002] und [TZITZIKAS ET AL. 2005] wird das formale Konzept eines Mediatormodells zum Aufbau eines komplexen Informationsnetzwerkes beschrieben. Als Beispiel eines solchen Netzwerkes wird das World-Wide-Web angeführt, in dem auf verschiedenen Rechnern und Datenbanken verteilt Informationen gespeichert sind, die vom Anwender abgerufen werden können. Das Finden der gewünschten Informationen ist dabei sehr zeitaufwendig, da die Stichwortsuche in einer derart großen Informationsmenge zu unpräzise ist. Strukturierte und gepflegte Wörterbücher (Ontologien) erleichtern die Informationssuche und liefern bessere Suchergebnisse, da hier die Informationen bereits vorsortiert sind. Allerdings können diese Wörterbücher unterschiedlich strukturiert sein und Informationen unter verschiedenen Begriffen ablegen bzw. indizieren. Der Anwender muss sich demnach mit den unterschiedlichen Strukturen und Begrifflichkeiten der einzelnen Wörterbücher auseinandersetzen, um die richtigen Informationen finden zu können. Als Lösung für diese Problematik wird ein Mediatormodell entwickelt und hergeleitet. Dieses Modell ermöglicht es, die Suche nach Begriffen in den einzelnen Wörterbüchern zu vereinheitlichen und die gewünschten Informationen zu finden.

Im Rahmen dieser Arbeit wird das Mediatormodell aus [TZITZIKAS ET AL. 2001] als Grundlage des Metadaten-Management-Systems verwendet und an die Belange der virtuellen Werkzeugmaschine als integrierten Entwicklungsarbeitsplatz angepasst. Der Schwerpunkt liegt in der Integration der verschiedenen Softwarewerkzeuge der einzelnen Fachbereiche. Die Aufgabe besteht im Wesentlichen darin, ein gemeinsames Wörterbuch für die Werkzeugmaschinenentwicklung aus Sicht der Datenintegration aufzubauen. Das Wörterbuch setzt sich aus den Begriffen der Modellierungstechniken zusammen, die in den jeweiligen Fachbereichen eingesetzt werden. Die Anpassungen des Mediatormodells aus [TZITZIKAS ET AL. 2001] betreffen vor allem den Aufbau des Wörterbuchs und die Suchfunktionen zum Auffinden bestimmter Datensätze.

Im Folgenden werden zunächst die zu integrierenden Softwarewerkzeuge formal als abstrakte Datenquellen eingeführt und erläutert. Diese sind im Wesentlichen als Datenbasen zu betrachten, die Daten unter bestimmten Begriffen abspeichern und dem Anwender eine einfache Sprache für das Finden ausgewählter Daten (Datenanfragen) zur Verfügung stellen. Im Anschluss daran wird das grundlegende Konzept des Mediators nach [TZITZIKAS ET AL. 2001]

eingeführt und dessen Funktion als Integrator verschiedener Datenquellen erläutert. Abschließend wird das Mediatormodell des Metadaten-Management-Systems vorgestellt, das sich von diesen beiden Konzepten ableitet.

4.5.1 Softwarewerkzeuge als abstrakte Datenquellen

Zur präzisen, unmissverständlichen Beschreibung technischer Details werden Normen und Richtlinien eingesetzt. Dadurch wird für einen bestimmten Bereich bzw. Arbeitsschwerpunkt eine Sprachregelung vorgeschrieben, die eine vereinheitlichte Begriffswelt (Vokabular) und Symbolik umfasst. So sind beispielsweise in der Norm IEC 61131-3 die SPS-Programmiersprachen definiert. Neben allgemeinen Begriffsbestimmungen werden die Befehle, Funktionen und Symbole der einzelnen Sprachen genau festgelegt. In der VDI-Richtlinie VDI 3260 geschieht dies für die Funktionsdiagramme, welche Zustände und Zustandsänderungen von Produktionsmaschinen graphisch darstellen, und in DIN ISO 1219 werden die Schaltzeichen und deren Semantik für die Entwicklung pneumatischer und hydraulischer Systeme definiert. Konsequenterweise folgen die eingesetzten Softwarewerkzeuge der in der Norm getroffenen Sprachregelung, indem sie die dort definierten Begriffe und Symbole für die Bezeichnung von Menüpunkten und Programmfunktionen verwenden und somit das Erstellen und Suchen von Daten dem Erfahrungskontext des Anwenders anpassen. Softwarewerkzeuge können folglich abstrakt als Datenquellen betrachtet werden. Im Folgenden wird daher eine formale Definition von Datenquellen eingeführt, die als Basis zur Beschreibung des Funktionsprinzips des im Rahmen der vorliegenden Arbeit entwickelten Metadaten-Management-Systems dient.

Nach [TZITZIKAS ET AL. 2001] und den zuvor aufgeführten Sachverhalten lassen sich Datenquellen aus einer Ontologie und einer Datenbasis (siehe Abbildung 4-5) aufbauen.

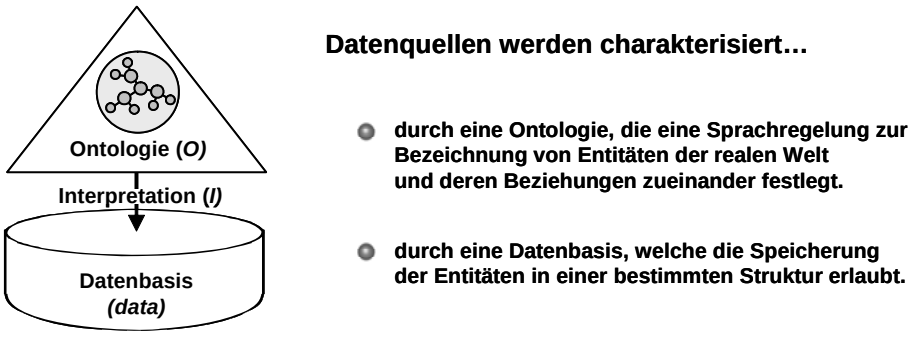


Abbildung 4-5: Softwarewerkzeuge als Datenquelle in der virtuellen Werkzeugmaschine

Die Ontologie entspricht dabei dem verwendeten Vokabular des Anwendungsbereichs einer Datenquelle in Form einer endlichen Menge von Termen (Terminologie) und einer darauf aufbauend definierten Zuordnungs- und Äquivalenzrelation. Die in der Terminologie enthaltenen Begriffe sind einer Norm oder Richtlinie entnommen und bezeichnen Entitäten aus dem Arbeitsumfeld des Anwenders oder eines Fachbereichs. Die Terminologie bildet also im Wesentlichen das notwendige, fachspezifische Vokabular zur Beschreibung der Daten ab. Die Zuordnungsrelation setzt die einzelnen Begriffe der Terminologie entsprechend ihres fachli-

chen Kontextes zueinander in Beziehung (siehe auch Abschnitt 2.2.2). Sie beschreibt eine partielle Ordnung dieser Begriffe und impliziert damit eine Klassifizierung bzw. Hierarchisierung derselben. Haben zwei unterschiedliche Begriffe aus der Terminologie dieselbe Bedeutung, so wird dies durch eine Äquivalenzrelation zwischen diesen beiden Begriffen gekennzeichnet. Die Daten der Datenquelle werden in der Datenbasis nach einer dort festgelegten Struktur und unter den in der Ontologie definierten Begriffen gespeichert. Die folgenden Definitionen geben die Erläuterungen formal wider:

Definition 4-2: Ontologie *([TZITZIKAS ET AL. 2001])***:**

Die Ontologie einer Datenquelle ist ein Tupel der Form $O = (T, \preceq, \sim)$ wobei

- T die Terminologie,
- $\preceq$ die *Zuordnungsrelation* mit $\preceq \subseteq T \times T$ und
- $\sim$ die *Äquivalenzrelation* mit $\sim \subseteq T \times T$ bezeichnet.

Die Ontologie ist eine Menge von Termen, die durch eine definierte Relation einander zugeordnet werden. Der Begriff der *Ontologie* ist grundlegend für die Definition der Datenquelle.

Definition 4-3: Datenquelle *([TZITZIKAS ET AL. 2001])***:**

Eine Datenquelle ist ein Tupel der Form $D_q = (O, data, I)$ wobei

- die Ontologie,
- *data* die Menge der Daten und
- I die *surjektive* Interpretationsfunktion mit $I : T \rightarrow 2^{data}$ bezeichnet.

Die Elemente der Menge *data* sind die in der Datenbasis gespeicherten Daten. Die Interpretationsfunktion I hingegen bildet einzelne Begriffe t aus der Terminologie T auf eine Teilmenge der Datenbasis ab. Dies bedeutet, dass der Zugriff auf die in der Menge *data* enthaltenen Daten anhand der in T definierten Begriffe erfolgt. Da I surjektiv ist, ist jedes Datum aus *data* mindestens einem Begriff t aus der Terminologie T zugeordnet.

Für einen Zugriff auf ausgewählte Inhalte einer Datenquelle sind weiterhin Abfragemechanismen notwendig. Eine hierfür geeignete Anfrage kann entsprechend Definition 4-4 formuliert werden.

Definition 4-4: Datenanfrage *([TZITZIKAS ET AL. 2002])***:**

Sei T die Terminologie einer Datenquelle und t ein Element $t \in T$ derselben, so gilt: Eine Datenanfrage q ist eine rekursive, mittels logischer Operatoren (und, oder, nicht, usw.) verkettete Menge von Termen mit folgendem Aufbau:

$$q ::= t \mid q \wedge q \mid q \vee q \mid q \wedge \neg q \mid (q) \mid \varepsilon$$

Eine Suchanfrage nach bestimmten Daten einer Datenquelle kann demnach ein Begriff aus der jeweiligen Terminologie oder eine Kombination aus diesen Begriffen sein. Aus Gründen der Vollständigkeit bezeichnet das Element ε die "leere" Datenanfrage.

Zur Verdeutlichung der beschriebenen Sachverhalte ist in Abbildung 4-6 ein 3D-CAD-Softwarewerkzeug als abstrakte Datenquelle dargestellt. Die Zuordnungsrelation $\preceq$ ist dabei mit einer durchgezogenen, die Interpretationsfunktion *I* mit einer gestrichelten Linie gekennzeichnet. Die Abbildung zeigt ein einfaches 3D-CAD-Modell, in welchem ein Ausschnitt einer Werkzeugmaschine zu sehen ist. Im Modell sind die Werkzeugwechsel-Einrichtung als „Assembly“ sowie der Maschinenständer und der Endschalter für die X-Achse als „Part“ enthalten. Der Endschalter wird als eine für die Softwareentwicklung relevante Komponente kategorisiert. Die Terminologie bezogen auf das aufgeführte Beispiel besteht aus den Elementen

$$T = \{\text{3D-CAD-Modell, Assembly, Part, Software\_relevante\_Komponenten}\}$$

und die Datenbasis aus den CAD-Modell-Elementen

$$data = \{\text{Werkzeugwechsel-Einrichtung, Maschinenständer, Endschalter Achse X}\}.$$

Für den Begriff *Assembly* als Parameter der Interpretationsfunktion *I* gilt beispielsweise *I(Assembly) = {Werkzeugwechsel-Einrichtung}.*

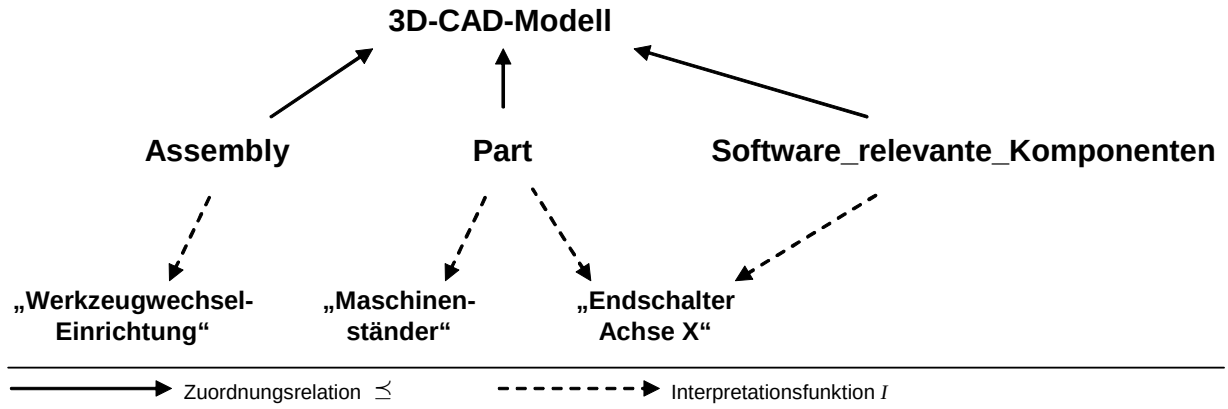


Abbildung 4-6: Darstellung eines 3D-CAD-Softwarewerkzeuges als abstrakte Datenquelle

Für die Suche nach allen in der Datenquelle enthaltenen Parts wird nach Definition 4-4 die Anfrage mit *q=Part* formuliert. Als Ergebnis liefert die Datenquelle die Elemente *Maschinenständer* und *Endschalter Achse X* zurück. Die Anfrage kann weiter detailliert werden, um gezielt nach einzelnen Daten suchen zu können. So gibt die Anfrage $q = Part \wedge Software\_relevante\_Komponenten$ als Ergebnis das Element *Endschalter Achse X* zurück.

Die Interpretationsfunktion *I* kann also in Bezug auf die Interpretation komplexer Datenanfragen wie folgt erweitert werden:

$$I(q \wedge q') = I(q) \cap I(q'),\ I(q \vee q') = I(q) \cup I(q') \text{ und } I(\neg q) = data \setminus I(q).$$

Die Antwort der Datenquelle auf eine Datenanfrage basiert demnach auf den einzelnen Interpretationsfunktionen, welche für die Terminologie *T* definiert sind. Damit eine solche Antwort richtige bzw. sinnvolle Daten zurückliefert, muss die Interpretationsfunktion, die zur Beantwortung der Anfrage verwendet wird, die Struktur der Ontologie berücksichtigen. Sie muss also konform zur Zuordnungsrelation sein. Im beschriebenen Beispiel sind die Begriffe *Assembly* und *Part* mit Daten aus der Menge *data* verknüpft. Der Begriff *3D-CAD-Modell* enthält keine Daten, allerdings sind ihm die Begriffe *Assembly* und *Part* zugeordnet. Um die von den Zuordnungsrelationen eingeführte partielle Ordnung der Terminologie einzuhalten, wird die Interpretationsfunktion für den Begriff *3D-CAD-Modell* als Vereinigungsmenge der Daten von *Assembly* und *Part* definiert. Formal lässt sich dies wie folgt beschreiben:

Definition 4-5: Modell einer Ontologie *([TZITZIKAS ET AL. 2002], [TZITZIKAS ET AL. 2004])***:**

Eine Interpretation *I* ist ein Modell einer Ontologie $O=(T,\preceq,\sim)$, wenn für alle $t,t'\in T$ bei $t\preceq t'$ die Aussage $I(t)\subseteq I(t')$ gilt. Für eine Interpretation *I* ist ein Modell $\bar{I}$ gegeben durch $\bar{I}(t)=\bigcup\{I(s);\, s\preceq t\},$ für $s,t\in T.$

Eine Datenanfrage *q* an die Datenquelle $D_q=(O,data,I)$ ist nach Definition 4-5 das Modell $\bar{I}(q)$ für D_q. Dadurch ist sichergestellt, dass die Anfrage *q* nach der in Definition 4-4 festgelegten Struktur genau die Daten als Ergebnis zurückliefert, die durch die Interpretationsfunktion *I* mit den entsprechenden Begriffen verknüpft sind. Die Definition 4-5 stellt somit die Konsistenz der Ontologie-Struktur sicher und bildet die Grundlage für das Verwalten und Wiederfinden von Informationen.

Für die Suche nach Daten in mehreren, verteilten Datenquellen müssen letztere in einer geeigneten Art und Weise miteinander verknüpft werden. Im folgenden Abschnitt wird eine Mediatorkomponente erläutert, welche die hierzu notwendigen Funktionen zur Verfügung stellt.

4.5.2 Integration der Datenquellen durch eine Mediatorkomponente

Ziel einer Lösung zur Datenintegration ist es, einen einheitlichen und konsistenten Zugriff auf die verschiedenen Datenquellen zu ermöglichen. Die Aufgabe besteht darin, in einer geeigneten Art und Weise die Ontologien der verschiedenen Datenquellen miteinander in Beziehung zu setzen. Damit erhält man eine vereinheitlichte Terminologie, welche die Grundlage für eine Anfragesprache nach Definition 4-4 bildet und den Zugriff auf eine Vielzahl von Datenquellen ermöglicht.

Das Metadaten-Management-System basiert auf dem Konzept der Mediation. Wie in Abschnitt 2.2.3 bereits erwähnt, ist ein Mediator eine Softwarekomponente, die eine Vermittlerfunktion zwischen mehreren Softwarewerkzeugen realisiert. Im Folgenden ist die Definition eines Mediators nach [TZITZIKAS ET AL. 2001] aufgeführt:

Definition 4-6: Mediator *([TZITZIKAS ET AL. 2001])*:

Ein Mediator M von k Datenquellen $\overset{k}{\underset{i=1}{D_{qi}}} = (O_i, data_i, I_i)$ besteht aus

- einer für alle Datenquellen gültigen, übergeordneten Ontologie $O_M = (T_M, \preceq, \sim)$ und
- einer Menge von Artikulationen a_i für jede Datenquelle D_{qi}, wobei jede Artikulation a_i eine Zuordnungsrelation zwischen der Terminologie der übergeordneten Ontologie und der Terminologie der Datenquelle $(T_M \cup T_i)$ darstellt.

Der Mediator M wird somit durch das Tupel $M = \langle (T_M, \preceq, \sim), a_1, ..., a_k \rangle$ für k Datenquellen $D_{q1}, ..., D_{qk}$ beschrieben.

Der Mediator nach Definition 4-6 hat einen ähnlichen Aufbau wie die in Definition 4-3 beschriebene Datenquelle. Der wesentliche Unterschied liegt darin, dass der Mediator selbst keine Daten speichert und somit auch keine Interpretationsfunktion I besitzt. Stattdessen enthält der Mediator eine Menge von Artikulationen, welche die Begriffe aus der Ontologie des Mediators mit jenen der einzelnen Datenquellen in Relation setzt. Suchanfragen an die Datenquellen werden vom Mediator über die in den Artikulationen definierten Verbindungen an die Datenquellen weitergeleitet. In Abbildung 4-7 ist das Beispiel eines Mediators für die Bereiche Mechanik- und Elektrokonstruktion sowie Softwareentwicklung zu sehen.

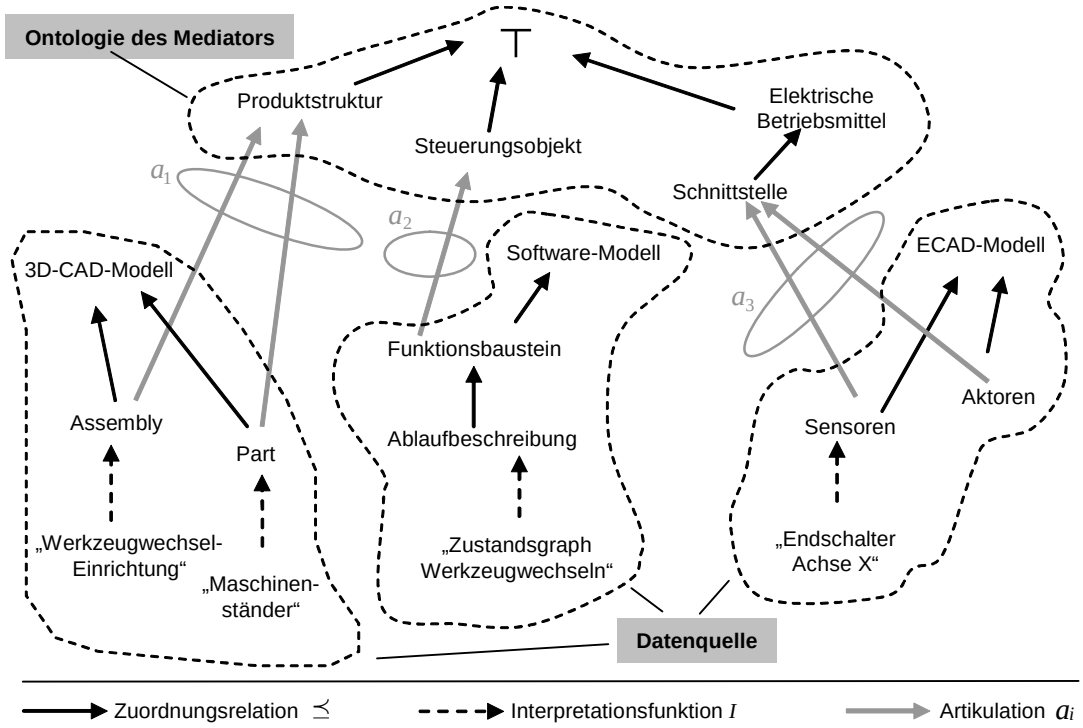


Abbildung 4-7: Ein Mediator für die Fachbereiche Mechanik-, Elektrokonstruktion und Softwareentwicklung

Das Beispiel zeigt einen Mediator für die drei Datenquellen D_{q1} = "3D-CAD-Modell", D_{q2} = "Software-Modell" und D_{q3} = "ECAD-Modell". Für die Artikulationen ergeben sich folgende Mengen:

- a_1 = {Assembly $\preceq$ Produktstruktur, Part $\preceq$ Produktstruktur}
- a_2 = {Funktionsbaustein $\preceq$ Steuerungsobjekt}
- a_3 = {Sensoren $\preceq$ Schnittstelle, Aktoren $\preceq$ Schnittstelle}

Durch die Artikulationen a_i wird eine Verbindung zwischen den Begriffen der Terminologie des Mediators T_M und jenen der Datenquellen T_i hergestellt. Dabei kann es vorkommen, dass in mehreren Datenquellen ein und derselbe Begriff verwendet wird, wobei solche Begriffe durchaus unterschiedliche Bedeutungen innerhalb der jeweiligen Datenquelle besitzen können. In diesem Fall spricht man von *Homonymen*. Das Gegenteil hiervon ist das *Synonym*, bei dem unterschiedliche Bezeichnungen für denselben Sachverhalt verwendet werden. Damit hinsichtlich dieser beiden Spezialfälle ein einwandfreies Funktionieren des Mediators gewährleistet werden kann, gelten folgende Annahmen: Für jedes $i \neq j$ ist $T_i \cap T_j = \varnothing$ und für jedes i gilt $T_M \cap T_i = \varnothing$. Dadurch werden Homonyme ausgeschlossen, da einzelne Begriffe in den verschiedenen Terminologien nur einmal vorkommen dürfen. Darüber hinaus sind zwei Begriffe t_i und t_j mit $t_i \in T_i, t_j \in T_j$ äquivalent, wenn für die beiden Artikulationen a_i, a_j und für einen Begriff $t \in T$ die Aussagen $t \sim_{a_i} t_i$ und $t \sim_{a_j} t_j$ gelten, d. h. wenn sie durch die Artikulationsbeziehungen als äquivalent dargestellt werden können. Anhand dieser Festlegung können Synonyme durch eine Äquivalenzrelation beschrieben und entsprechend zugeordnet werden.

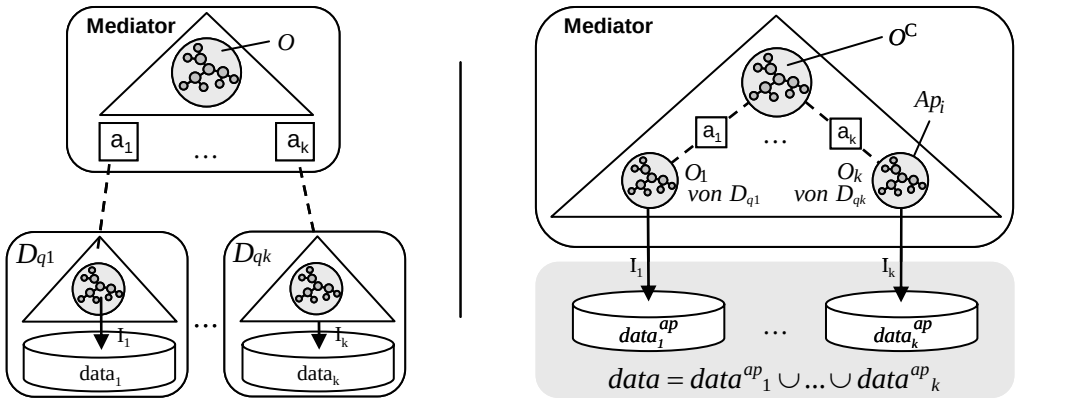
Für die weitere Anwendung des beschriebenen Mediatorkonzepts im Rahmen des Metadaten-Management-Systems kann von vereinfachenden Voraussetzungen ausgegangen werden. Diese ergeben sich durch die verschiedenartigen Anforderungen an eine Mediatorkomponente in Bezug auf deren Anwendungsbereich. Im folgenden Abschnitt werden diese vereinfachenden Annahmen motiviert und erläutert sowie in einer für das Metadaten-Management-System spezifischen Definition eines Mediators zusammengeführt.

4.5.3 Die Mediatorkomponente des Metadaten-Management-Systems

In der Arbeit von [TZITZIKAS 2002] wird das Mediatorkonzept dazu verwendet, einen einheitlichen Zugriff auf eine Vielzahl unterschiedlich strukturierter Informationsquellen zu ermöglichen (siehe Abbildung 4-8 a). Bei den Informationsquellen handelt es sich vorwiegend um Online-Kataloge oder Datenbanken aus gleichen Anwendungsbereichen [TZITZIKAS ET AL. 2001]. Der Schwerpunkt der Arbeiten liegt in der Verknüpfung der Informationsquellen, um eine einheitliche Datenabfragesprache zu ermöglichen, sowie in der Definition geeigneter Suchstrategien, welche die Qualität der Stichwortsuche im Mediator steuern können. Dadurch ist es beispielsweise möglich, die Suche eng an die angegebenen Stichwörter zu binden oder auf verwandte Begriffe zu erweitern, um die Menge der Suchergebnisse zu vergrößern.

Im Zusammenhang mit dem Metadaten-Management-System wird das Konzept der Mediation dazu verwendet, verschiedene Softwaresysteme bzw. vielmehr deren Modellierungssprachen und -techniken miteinander zu verknüpfen. Dabei spielt die Implementierung verschiedener Anfrage- und Suchstrategien eine untergeordnete Rolle. Vielmehr liegt die Aufgabe darin, die in den Dokumenten der Softwarewerkzeuge enthaltenen Daten und deren Semantik auf einer abstrakten Metaebene miteinander zu verbinden, um eine interdisziplinäre Konsistenzsicherung zu ermöglichen.

In Abbildung 4-8 b ist das von [TZITZIKAS 2002] abgeleitete Mediatormodell für die virtuelle Werkzeugmaschine zu sehen. Die Ontologie setzt sich hierbei aus zwei Teilen zusammen. Der Kern O^c definiert fundamentale Begriffe und bildet somit eine Basisterminologie, die für alle Fachbereiche zutreffend ist. Allgemein gültige Begriffe wie *Funktion* oder *Komponente* werden demnach im Kern definiert.



a) Mediator nach Tzitzikas b) Mediator des Metadaten-Management-Systems

Abbildung 4-8: Vereinfachtes Mediatormodell für das Metadaten-Management-System [ZAEH & LERCHER 2006]

Des Weiteren sind in der Ontologie des Mediators so genannte Applikationsmuster A_{pi} enthalten. Sie bilden all jene Datenstrukturen ab, die im Informationsverbund der virtuellen Werkzeugmaschine zwischen den Fachbereichen ausgetauscht werden (relevante Informationen, siehe Abschnitt 4.1). Für jedes Softwarewerkzeug bzw. Modell, das in die virtuelle Werkzeugmaschine integriert wird, ist im Mediator ein entsprechendes Applikationsmuster zu definieren. Durch die Artikulationsbeziehungen werden diese mit dem Kern und damit implizit miteinander verknüpft.

Die beiden Mediatormodelle in Abbildung 4-8 spiegeln die verschiedenen Anwendungsbereiche und Ausgangsvoraussetzungen wider. Der Mediator in Abbildung 4-8 a dient dazu, verschiedene, vorgegebene Datenbanken miteinander zu verbinden und einen einheitlichen Datenzugriff über eine gemeinsame Datenabfragesprache zu ermöglichen. Die Ontologien der einzelnen Datenquellen sind dabei fest vorgegeben und werden anhand von Artikulationsbeziehungen mit der Ontologie des Mediators verbunden. Im Gegensatz dazu sind bei dem Mediatormodell aus Abbildung 4-8 b die Ontologien der zu integrierenden Softwarewerkzeuge nicht explizit vorgegeben, sondern Bestandteil des Mediators selbst. Sie dienen als abstrakte Beschreibung der Datenstrukturen und -beziehungen der einzelnen Softwarewerkzeuge. Die

Applikationsmuster stellen ein Abbild der Softwarewerkzeuge im Mediator dar und werden über den im Kern definierten Grundwortschatz miteinander in Beziehung gesetzt. Beim Entwurf des Mediators sind somit die Strukturen und Begriffe des Kerns und der Applikationsmuster aufeinander abzustimmen. Der Vorteil dieses Aufbaus liegt vor allem darin, dass die Menge der Artikulationsbeziehungen kompakt gehalten werden kann und somit eine vereinfachte Datenabfrage möglich ist. Die Interpretationsfunktion *I* ist für die Ontologien der Applikationsmuster definiert. Die Datenobjekte der Menge *data* werden im Gegensatz zum Mediatorkonzept aus Abbildung 4-8 a in einer einzigen, gemeinsamen Datenbank gespeichert. Sie umfassen die relevanten Daten, die zwischen den Fachbereichen ausgetauscht werden (vergleiche Definition 4-1).

Die in der Ontologie des Mediators definierten Begriffe bezeichnen Elemente fachspezifischer Modellierungstechniken, beispielsweise aus dem CAD-Bereich oder der SPS-Softwareentwicklung. Die Eigenschaften dieser Modellelemente lassen sich durch die Angabe von Attributen näher beschreiben. Betriebsmittelkennzahlen sind beispielsweise Attribute von CAD-Komponenten. Attribute können als eigene Begriffe von der Terminologie *T* eingeführt und durch eine Interpretationsfunktion mit den Datenobjekten in Relation gesetzt werden. Dies hat jedoch zwei Nachteile. Zum einen muss für jede neue Betriebsmittelkennzahl die Ontologie erweitert werden und zum anderen wird diese mit der Zeit in ihrer Größe unüberschaubar. Für den Mediator des Metadaten-Management-Systems ist es deshalb erforderlich, dass den Begriffen der Mediator-Ontologie direkt Attribute zugewiesen werden können. Die Attribute charakterisieren Eigenschaften jener Datenobjekte, die den jeweiligen Begriffen über die Interpretationsfunktion zugewiesen sind. In der folgenden Definition des Mediatorkonzepts für das Metadaten-Management-System sind diese Sachverhalte formal beschrieben:

Definition 4-7: Mediator des Metadaten-Management-Systems:

Der Mediator ist durch das Tupel $M^{vmt} = \langle (O^c, a_1, ..., a_k), T^{vmt}, Ap_1, ..., Ap_k, P, \nu, ref, data \rangle$ gegeben, wobei

- O^c die Ontologie des Kerns mit $O^c = (T^c, \preceq^c, \sim^c)$ bezeichnet und $a_1, ..., a_k$ die Artikulationsbeziehungen zwischen O^c und *k* Applikationsmustern darstellt,
- T^{vmt} die Terminologie des Mediators ist mit $T^{vmt} = T^c \cup T_i^{ap}$, für i=1,..., k,
- Ap_i die Applikationsmuster sind, mit $Ap_i = \langle (T_i^{ap}, \preceq^{ap}, \sim^{ap}), I^{ap}, data^{ap} \rangle$, für i=1,..., k,
- *P* die Attributmenge ist,
- ν die Attributfunktion mit $\nu : P \to T^{vmt}$ ist,
- *ref* die Referenzrelation mit $ref \subseteq T^{vmt} \times T^{vmt}$ bezeichnet und
- *data* die Menge der Datenobjekte darstellt.

Die Ontologie des Mediators M^{vmt} ergibt sich somit durch das Tupel $O^{vmt} = (T^{vmt}, \preceq^c, \sim^c, \preceq_i^{ap}, \sim_i^{ap}, a_i)$, für i=1,..., k.

Die Implementierung des Metadaten-Management-Systems basiert auf dem in Definition 4-7 beschriebenen Mediatormodell. Die Terminologie T^{vmt} des Mediators ist die Vereinigungsmenge aus der Terminologie T^c des Kerns und den Terminologien T_i^{ap} der einzelnen Applikationsmuster. Die Elemente der Attributmenge *P* sind so genannte *Literale*. Literale sind unveränderliche Werte, wie beispielsweise die Zahl *2* oder die Zeichenkette *Name* [KEMPER & EICKLER 1997].

Durch die Attributfunktion ν können den Begriffen aus T^{vmt} Attribute zugewiesen werden. Beispielsweise werden die Baugruppen und Komponenten einer CAD-Produktstruktur zur eindeutigen Kennzeichnung mit einer Betriebsmittelkennzahl versehen. Die Attributfunktion lässt sich hierfür in der Form $\nu(Betriebsmittelkennzahl) = \{Part, Assembly\}$ beschreiben. Durch die Funktion ν können charakteristische Eigenschaften oder Merkmale der Datenobjekte den Begriffen zugewiesen werden, denen sie mittels der Interpretationsfunktion *I* zugeordnet sind. Der Zugriff auf ein Attribut wird durch einen Punkt gekennzeichnet, also in der Form *t.p* mit $t \in T^{vmt}$ und $p \in P$. Eine Datenanfrage nach Definition 4-4 kann nun durch eine Suchanfrage in der Form $q=Part.Betriebsmittelkennzahl=BMK\text{-}3411\text{-}AX$ formuliert werden.

Die in T^{vmt} enthaltenen Begriffe bezeichnen Modellelemente aus den verschiedenen Fachbereichen. So sind hier die Begriffe *Funktionsbaustein* aus der Steuerungsentwicklung oder *Part* und *Assembly* aus der Mechanikkonstruktion in T^{vmt} enthalten. Um dem Mediator die Möglichkeit zu geben, fachbereichsübergreifende Konsistenzbedingungen zu prüfen, sind zwischen den Begriffen aus T^{vmt} fachlich relevante Beziehungen zu definieren. Solche Beziehungen werden durch die Referenzrelation *ref* beschrieben. Beispielsweise kann definiert werden, dass ein Part oder ein Assembly durch einen Funktionsbaustein der SPS-Software gesteuert wird. Eine entsprechende Relation lässt sich dann durch die Schreibweise $(Part, Funktionsbaustein) \in ref$ abbilden. Während die Zuordnungsrelationen $\preceq^c$ des Kerns bzw. $\preceq_i^{ap}$ der Applikationsmuster eine taxonomische Hierarchie der Begriffe aus T^{vmt} definieren, beschreibt die Referenzrelation *ref* technische Beziehungen zwischen diesen Begriffen. Die Referenzrelation *ref* legt damit den so genannten Kontext eines Begriffs fest. Das heißt, $\forall r \in ref$ gilt $r \notin \preceq^c$, $r \notin \sim^c$, $r \notin \preceq_i^{ap}$ und $r \notin \sim_i^{ap}$.

Eine weitere Änderung gegenüber dem Mediatorkonzept aus Definition 4-6 betrifft die Artikulationsbeziehungen zwischen dem Kern und den Applikationsmustern. Der Kern beschreibt einen Grundwortschatz von Begriffen, der die Applikationsmuster miteinander verbindet. Daraus folgt, dass die Begriffe aus der Terminologie der Applikationsmuster stets den Begriffen aus der Terminologie des Kerns zugewiesen sind und nicht umgekehrt. Es gilt also, wenn $t\, a_i\, t'$, dann ist $t \in T_i^{ap}$ und $t' \in T^c$. Für das Mediatormodell in [TZITZIKAS ET AL. 2001] ist diese Vereinfachung nicht möglich, da dort die ontologischen Strukturen der Datenquellen nicht beeinflusst werden können und somit die Einordnung der Begriffe der Datenquellen in den Mediator den Vorgaben einer effizienten Suchstrategie unterliegt.

4.6 Zusammenfassung

Die verschiedenen Fachbereiche verwenden bislang unterschiedliche Begriffe, um die Entitäten ihres Fachgebiets zu bezeichnen. Die Bedeutung der Begriffe kann selbst in den einzelnen Fachbereichen variieren, was letztendlich zu Missverständnissen führt. Mit den verwendeten Begriffen wird demnach eine fachbereichsspezifische Semantik impliziert, deren Kenntnis die

Grundlage einer interdisziplinären Zusammenarbeit darstellt. Das vorgestellte Mediatormodell greift diesen Sachverhalt auf und beschreibt ein Softwarewerkzeug als so genanntes Applikationsmuster. Dieses besteht aus einem ontologischen Modell zur Abbildung der Semantik sowie einer Interpretationsfunktion, welche die Begriffe der Ontologie auf die Daten der Datenbasis des Mediatormodells und damit auf die Entwicklungsdaten der eingesetzten Entwicklungswerkzeuge abbildet. Der Mediator selbst verwendet ein ontologisches Modell, um einen Grundwortschatz als gemeinsamen Integrationsrahmen für alle Fachbereiche zu definieren. Zusammen mit den Ontologien der Applikationsmuster entsteht somit eine semantische Basis, welche die fachbereichsübergreifenden Abhängigkeiten beschreibt. Die Formalisierung des in Definition 4-7 vorgestellten Mediatormodells erfolgt anhand eines Metamodells, das im folgenden Kapitel vorgestellt wird.

5 Das Machine-Tool-Metamodell

5.1 Übersicht

Das Integrationskonzept des Metadaten-Management-Systems basiert auf den im Metamodell definierten Entitäten. Deren grundlegende Strukturen und Abhängigkeitsbeziehungen sind im Mediator nach Definition 4-7 festgelegt. Das Metamodell stellt die Formalisierung der Ontologie des Mediators O^{vmt} und der Applikationsmuster Ap_i dar. Da die Struktur des Metamodells von den Gegebenheiten und Anforderungen des jeweiligen Unternehmens abhängt, muss dieses situationsbedingt angepasst werden. Aus diesem Grund wird zunächst ein Vorgehensmodell erläutert, das den methodischen Aufbau des Metamodells beschreibt und auf die dabei zu berücksichtigenden Einflussgrößen eingeht. Im Anschluss wird ein Konzept zur Funktionsmodellierung von Werkzeugmaschinen vorgestellt. Es folgt dem Paradigma des Model Driven Engineering und dient beispielhaft als Anwendungsszenario für die virtuelle Werkzeugmaschine. Insbesondere leiten sich daraus die Modellkonstrukte für das Metamodell ab, welches im anschließenden Abschnitt vorgestellt wird.

5.2 Methodik zum Aufbau des Metamodells

In der Kern-Ontologie O^c und den Applikationsmustern Ap_i werden die Modellelemente der verschiedenen Fachbereiche in einen gemeinsamen Kontext gesetzt. Durch die Referenzrelation *ref* werden Beziehungen zwischen verschiedenen Modellelementen aus M^{vmt} definiert. Ausschlaggebend hierfür ist das Wissen über die fachlichen Zusammenhänge und Abhängigkeiten, die sich bei der Entwicklung und Konstruktion einer Werkzeugmaschine ergeben. Ebenso ist eine fundierte Kenntnis der eingesetzten Modelltechniken und -strategien erforderlich, um die Strukturen der Mediator-Ontologie O^{vmt} entwickeln zu können. In den folgenden beiden Abschnitten wird daher erläutert, mit welchen Methoden der in Definition 4-7 vorgestellte Mediator aufgebaut wird. Dazu werden zunächst die Einflussgrößen beschrieben, die es bei der Einführung und dem Aufbau einer Integrationslösung zum Datenaustausch zu beachten gilt. Anschließend wird eine Vorgehensweise vorgestellt, um systematisch die einzelnen Softwarewerkzeuge in den Informationsverbund der virtuellen Werkzeugmaschine einzubinden.

5.2.1 Einflussgrößen

Für den Aufbau der Mediator-Ontologie nach Definition 4-7 sind zwei Schwerpunkte entscheidend. Zum einen ist die technische Verzahnung der einzelnen Bauteile und Baugruppen bei der Entwicklung und Konstruktion sowie die dafür notwendigen organisatorischen Abläufe zu analysieren. Dieser Aspekt geht der wesentlichen Frage nach, wie sich diese Abhängigkeiten in einer fachbereichsübergreifenden Modellierung widerspiegeln. Es gilt beispielsweise zu bestimmen, welche fachspezifischen Modellelemente und Strukturierungsmittel äquivalent sind und aufeinander abgebildet werden können bzw. welche Modellelemente sich ergänzen und verschiedene Aspekte eines Systems oder einer Komponente beschreiben. In der Ontologie werden diese Modellelemente hierarchisch gegliedert und entsprechend ihrer fachlichen Abhängigkeiten miteinander in Beziehung gesetzt. Darüber hinaus wird festgehalten, welche

Informationen von anderen Fachbereichen benötigt werden, um ein Modell zu erstellen. Dies setzt eine Analyse der organisatorischen Abläufe und Geschäftsprozesse voraus.

Der zweite Schwerpunkt bei der Zusammensetzung der Mediator-Ontologie betrifft die Datenformate der eingesetzten Softwarewerkzeuge. Dabei ist von Interesse, wie der Aufbau und die Struktur der jeweiligen Formate definiert sind und in welcher Form die Daten abgespeichert werden. Folgt das Format einer nationalen oder internationalen Norm, kann deren Struktur häufig durch eine Grammatik dargestellt werden, die dann von einem Parser verarbeitet wird. Wesentlich dabei ist, unter welchen Begriffen die Daten und Informationen abgespeichert werden. Die in den Datenformaten benutzte Terminologie muss bei der Definition der Applikationsmuster berücksichtigt werden, um eine semantisch korrekte Einordnung in das Metamodell zu gewährleisten.

Bei der Betrachtung dieser beiden Schwerpunkte sind bestimmte Einflussgrößen [TRIPPNER 2002] zu beachten (siehe Tabelle 5-1), die für den Aufbau der virtuellen Werkzeugmaschine im Unternehmen eine grundlegende Rolle spielen.

	Betrachtung der technischen und organisatorischen Zusammenhänge	**Betrachtung der Datenmodelle und Modellierungstechniken**
Produkt	- Vielfalt der Komponenten - komplexe Produktfunktionen	Neue Modellierungstechniken zur Abbildung der Maschineneigenschaften
Produktentwicklungs-prozess	- neue Methoden - neue Softwarewerkzeuge - zusätzliche Kompetenzträger	- Schnittstellenproblematik - semantische Heterogenität
Produktdaten	- Vielfalt der Daten - Änderungshäufigkeit - Verantwortlichkeiten	- Standards und Normen - Datenformate
Projektmanagement	Steigende Effizienzanforderung	Einheitlicher Zugriff auf die Projektdaten

Tabelle 5-1: Einflussgrößen bei der Einführung der virtuellen Werkzeugmaschine

Produkt

Das Produkt *Werkzeugmaschine* stellt die bedeutendste Einflussgröße dar. Je nach Art und Komplexität sind verschiedene Modellierungstechniken zur Abbildung und Simulation der Maschineneigenschaften erforderlich. Neben einem mechanischen Funktionsträger beinhalten moderne Werkzeugmaschinen elektromechanische Komponenten, die durch die Steuerungssoftware in komplexe Maschinenabläufe eingebunden werden. Die Maschinenfunktionen werden damit durch das Zusammenwirken mechanischer, elektromechanischer und softwaretechnischer Komponenten realisiert. Um die Charakteristika dieser mechatronischen Bauweise in der Entwicklung berücksichtigen zu können, sind neue Beschreibungsmittel notwendig, welche die Interaktionen zwischen den einzelnen Komponenten konsistent und fachbereichsübergreifend abbilden. Beim Aufbau bzw. bei der Erweiterung der virtuellen Werkzeugmaschine ist dies zu berücksichtigen, indem die zentralen Modellelemente harmonisch in den gesamten Kontext des Metamodells eingefügt werden.

Produktentwicklungsprozess

Die Erweiterung des Metamodells erfordert nicht nur Kenntnisse über die Modellierungstechniken an sich, sondern insbesondere auch über deren methodische Anwendung im Rahmen der interdisziplinären Entwicklungsabläufe. Die organisatorischen Querbeziehungen zwischen den Fachbereichen spiegeln sich in den Beziehungen zwischen den Modellelementen des Metamodells wider. Diese Beziehungen ermöglichen es, die Konsistenz der Daten zu überprüfen sowie Daten und Informationen aus den einzelnen Fachbereichen zu neuen Entwicklungsdokumenten zusammenzustellen. Darüber hinaus erfordert es die Mechatronik gegebenenfalls, zusätzliche Kompetenzen in Form von neuen Fachbereichen oder Subunternehmen in die Entwicklung mit einzubeziehen. Deren Abläufe, Modellierungstechniken, Daten und Entwicklungsdokumente müssen ebenso von der virtuellen Werkzeugmaschine verarbeitet werden.

Produktdaten

Bei der Beschreibung und Simulation der Maschineneigenschaften werden verschiedene Daten und Informationen benötigt, die entsprechend dem Projektfortschritt in unterschiedlicher Granularität und Qualität vorliegen. Beim Aufbau der virtuellen Werkzeugmaschine ist deshalb präzise festzulegen, welche Daten zwischen den Fachbereichen ausgetauscht werden sollen und zu welchem Zweck. Hier gilt es, die verwendeten Datenformate, Standards und Normen zu untersuchen und gemeinsame Strukturen zu identifizieren. Hierzu sind geeignete Datenmodellierungskonzepte einzusetzen, welche die Vielschichtigkeit der abzubildenden Datenformate berücksichtigen. Insbesondere müssen die Systeme definiert werden, die vorrangig für die Spezifikation bestimmter Daten verantwortlich sind. Beispielsweise ist festzulegen, ob die Produktstruktur primär von einem CAD- oder ERP-System verwaltet wird.

Projektmanagement

Ein effizientes und wirkungsvolles Projektmanagement ist geprägt von einem schnellen und transparenten Zugriff auf aktuelle Daten und Informationen des laufenden Entwicklungsprojekts. Die virtuelle Werkzeugmaschine stellt eine Plattform zur Verfügung, die einen einheitlichen Zugang zu den von ihr verwalteten Daten sicherstellt. Hierzu sind geeignete Zugriffsmechanismen vorzusehen, die ein schnelles Auffinden der gewünschten Informationen ermöglichen und unabhängig von den zu durchsuchenden Datenstrukturen eine einheitliche Anwenderschnittstelle bieten.

5.2.2 Vorgehensweise

Um die verschiedenen Einflussgrößen beim Aufbau des Metamodells systematisch zu erfassen, wird die in Abbildung 5-1 dargestellte Vorgehensweise vorgeschlagen. Ziel ist es, die im Entwicklungsprozess eingesetzten Modellierungselemente und Datenformate in eine fachbereichsübergreifende Ontologie einzuordnen und auf einen einheitlichen Aggregationsgrad zu abstrahieren.

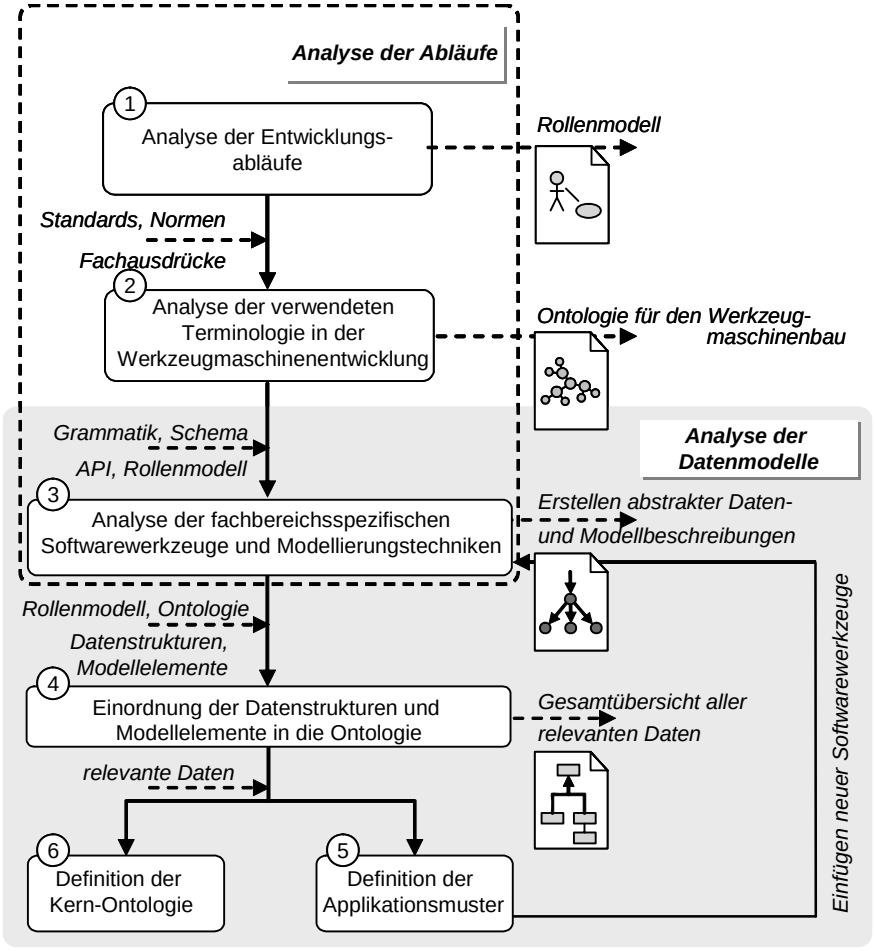


Abbildung 5-1: Vorgehensweise zur Definition des Metamodells

Um die Sachverhalte anschaulich vermitteln zu können, werden im Folgenden die einzelnen Schritte des Vorgehensmodells anhand eines durchgängigen Beispiels konkretisiert.

Analyse der Entwicklungsabläufe

Der erste Schritt beim Aufbau der virtuellen Werkzeugmaschine als zentrale Integrationslösung ist die genaue Analyse der Entwicklungsabläufe im Unternehmen. Dabei gilt es, die beteiligten Fachbereiche und deren Abstimmungsbedarf zu identifizieren und zu dokumentieren. Die Ergebnisse der Analyse werden in einem Rollenmodell zusammengefasst, das die einzelnen Aufgaben und die dafür verantwortlichen Rollen darstellt (siehe Abbildung 5-2). Die Aufgaben unterliegen dabei in einem ersten Schritt keiner strengen zeitlichen Reihung. Vielmehr erfolgt eine Gruppierung nach kausalen Zusammenhängen, um eine logische Abfolge der einzelnen Arbeitsschritte aufzeigen zu können.

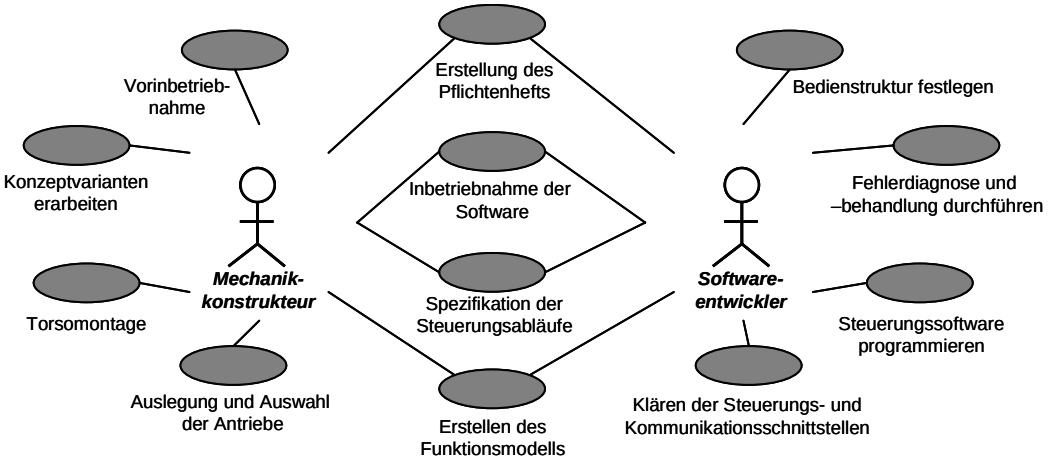


Abbildung 5-2: Beschreibung der Aktivitäten und Rollen

Das Rollenmodell gliedert sich im Wesentlichen in zwei Teile. Im ersten Abschnitt werden die Rollen und ihr Arbeitsumfeld dargestellt. Dabei ist jede Rolle einem Fachbereich oder einem Unternehmen zugeordnet. Die Erläuterung des Arbeitsumfelds umfasst eine kurze *Tätigkeitsbeschreibung*, die Angabe benötigter *Datenquellen* wie beispielsweise bestimmte Entwicklungsdokumente oder -modelle, eine Auflistung der *Interessensschwerpunkte* und der *Zielsetzung* der Tätigkeit sowie die Dokumentation der *Entwicklungsunterlagen* bzw. *-modelle*. Darüber hinaus werden alle Aufgaben festgehalten, an der die Rolle beteiligt ist. Der zweite Abschnitt ist in die einzelnen Entwicklungsphasen gegliedert und beschreibt im Detail die von den Rollen auszuführenden Aktivitäten. Für die Darstellung kommen *Use Case*- und *Sequenzdiagramme* aus der UML zur Anwendung. Die UML bietet mit diesen beiden Diagrammtypen eine einfache und übersichtliche Darstellungsmöglichkeit, die auch von Personen ohne Grundwissen in der Darstellung von Prozessmodellen interpretiert werden kann. Zusammen mit einer tabellarischen Beschreibung der Tätigkeiten, der Arbeitsgrundlagen (z. B. Dokumente, CAD-Zeichnungen oder Schaltpläne), der Interessensschwerpunkte sowie der Entwicklungsdokumente ermöglichen diese Diagramme eine übersichtliche Abbildung des Vorgehens bei der Entwicklung einer Werkzeugmaschine. Nach Abschluss der Analyse sind die im Unternehmen vorhandenen Fachbereiche sowie deren Aufgabenbereiche und Abläufe bekannt. Im nächsten Schritt kann nun detailliert auf die in den Fachbereichen verwendeten Fachsprachen eingegangen werden.

Analyse der verwendeten Terminologie in der Werkzeugmaschinenentwicklung

Die virtuelle Werkzeugmaschine stellt eine einheitliche Sicht auf die Entwicklungsdaten mehrerer, unterschiedlicher Datenquellen zur Verfügung. Anstatt viele Datenanfragen in verschiedenen spezifischen Anfragesprachen stellen zu müssen, ist in der virtuellen Werkzeugmaschine nur eine Anfrage, unabhängig von den darunter liegenden Datenquellen und Datenstrukturen, notwendig. Voraussetzung hierfür ist unter anderem eine Harmonisierung der verschiedenen Begriffswelten der Fachbereiche auf einer gemeinsamen Abstraktionsebene. Dies erfolgt durch die Definition der fachbereichsübergreifenden Ontologie O^{vmt}. Das Ziel der Begriffsbestimmung ist es, Herkunft und Semantik der Begriffe zu erfassen und sie in die hierarchische Gliederung der Terminologie einzuordnen (siehe Abbildung 5-3).

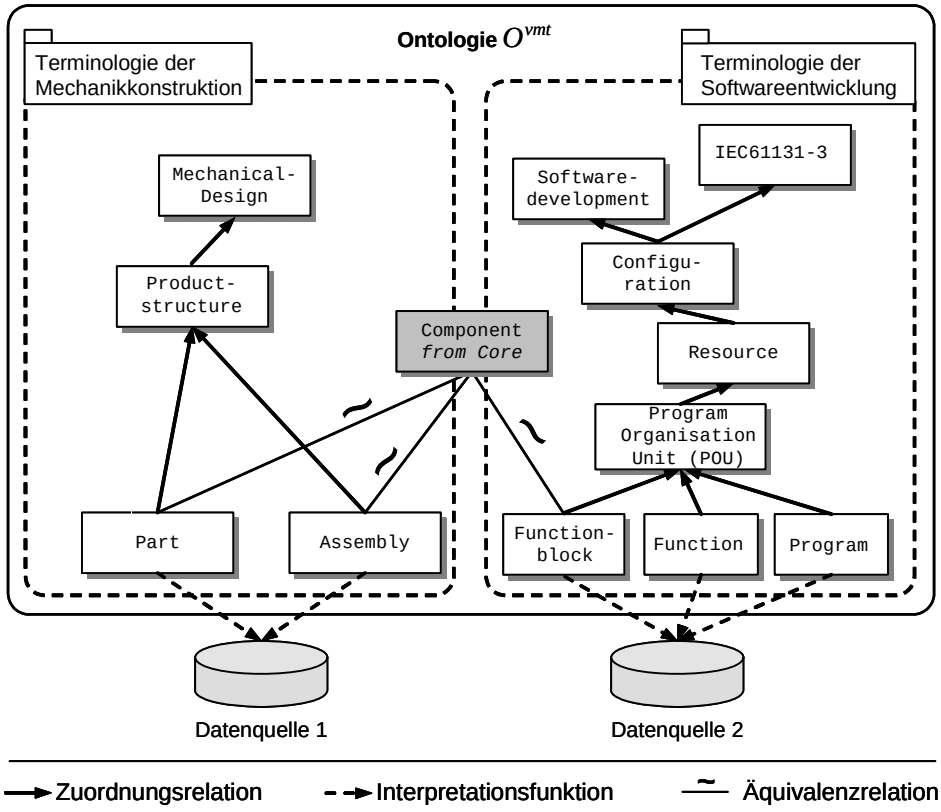


Abbildung 5-3: Einordnung der Begrifflichkeiten in eine Terminologie

Insbesondere sollen Synonyme und Homonyme herausgearbeitet und in der Ontologie entsprechend gekennzeichnet werden. Die Grundlage dieser Aufgabe stellen vorwiegend nationale und internationale Standards sowie firmeninterne Normen dar. Dort sind die Bezeichnungen und Strukturierungsmittel der Fachbereiche genau definiert.

Für die Dokumentation der Ontologie werden Klassendiagramme aus der UML verwendet. Dabei wird zunächst jeder Fachbereich als eigenständiger Namensraum anhand eines UML-Pakets abgebildet. Die Begriffe aus $t \in T^{vmt}$ werden als Klassen dargestellt und mit einer Generalisierungsbeziehung in eine erste grobe terminologische Hierarchie eingebunden. Allgemein gültige Begriffe, die nicht eindeutig einem Fachbereich zugeordnet werden können, werden in das Paket *Core* eingefügt. In Abbildung 5-3 wird der Begriff *Komponente* nicht nur für die Bezeichnung von mechanischen Bauteilen und Baugruppen verwendet, sondern auch für die Identifizierung von Softwarestrukturen. Dieser Begriff wird demnach in das Paket *Core* eingetragen und durch Äquivalenzrelationen mit den entsprechenden Begriffen verbunden.

Nach Beendigung dieses Schrittes liegt ein UML-Modell vor, das für jeden Fachbereich ein Klassendiagramm enthält, in dem die verwendeten Fachbegriffe hierarchisch eingeordnet sind. Fachbereichsübergreifende Begriffe werden in ein spezielles Paket eingefügt und durch Referenz- bzw. Äquivalenzrelationen mit den anderen Begriffen verknüpft.

Analyse der fachbereichsspezifischen Softwarewerkzeuge und Modellierungstechniken

Für jeden Fachbereich werden nun die eingesetzten Softwarewerkzeuge und Modellierungstechniken untersucht. Kern der Aufgabe ist es, die prinzipiellen Strukturen der Datenformate zur Speicherung der Modelle herauszuarbeiten und mit geeigneten Mitteln zu dokumentieren. Dies dient dazu, die Berührungspunkte der Datenformate zu erkennen und diese auf einer gemeinsamen Abstraktionsebene zusammenzuführen.

Zu diesem Zweck wird zunächst für jedes Softwarewerkzeug ein UML-Paket erstellt und mit einem eindeutigen Namen gekennzeichnet. Der Name ist ein Akronym der Bezeichnung des Softwarewerkzeuges oder der zu Grunde liegenden Modellierungstechnik. Dieses UML-Paket repräsentiert im Metamodell das Applikationsmuster, welches als Klassendiagramm spezifiziert wird. Die Analyse der Datenformate der Dokumente erfolgt auf der Grundlage von Standards bzw. Normen, in denen DTDs, Schemadefinitionen oder allgemeine Grammatiken formal beschrieben werden. Entscheidend ist hierbei die Frage, welche Daten von fachbereichsübergreifendem Interesse und demnach im Sinne der virtuellen Werkzeugmaschine als *relevant* zu bezeichnen sind. Die Informationen darüber enthält das Rollenmodell. Hier sind die Aktivitäten, die beteiligten Rollen sowie die erstellten und verwendeten Dokumente beschrieben. Damit lassen sich die Interaktionspunkte der Fachbereiche sowie deren Informationsaustausch untereinander ableiten. In Abbildung 5-4 ist beispielhaft das Szenario im Rahmen der Erstellung des Funktionsmodells einer Werkzeugmaschine dargestellt.

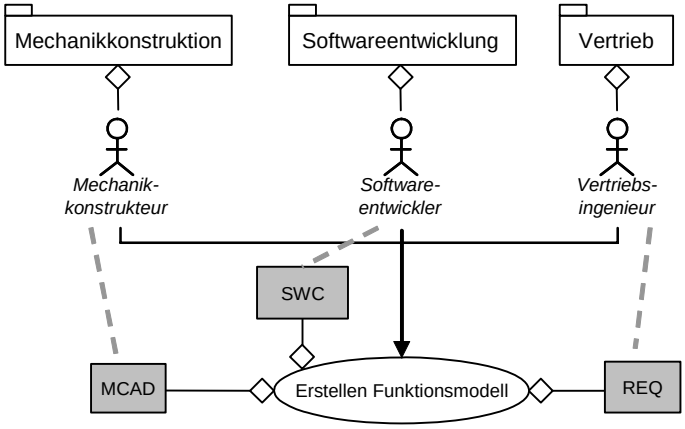
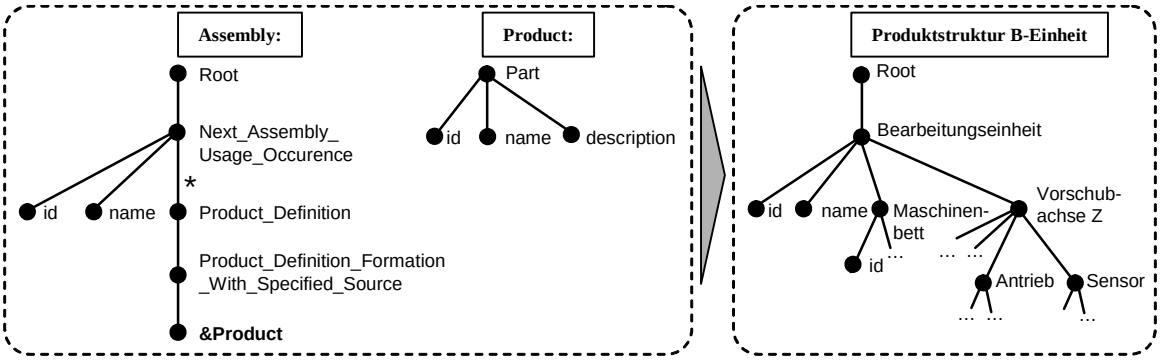


Abbildung 5-4: Ermitteln der relevanten Informationen zwischen den Fachbereichen

Den beteiligten Fachbereichen *Mechanikkonstruktion*, *Softwareentwicklung* und *Vertrieb* sind die Rollen *Mechanikkonstrukteur*, *Softwareentwickler* und *Vertriebsingenieur* zugeordnet. Diese arbeiten bei der Erstellung des Funktionsmodells der Werkzeugmaschine zusammen, wobei sie verschiedene, spezifische Softwarewerkzeuge einsetzen. In Abbildung 5-4 sind die Bezeichnungen der drei Applikationsmuster zu sehen. Die Bezeichnung *MCAD* steht hierbei für das System zur Erstellung von 3D-CAD-Modellen, *SWC* für das Werkzeug zur Beschreibung des Softwarekonzepts sowie *REQ* zur Aufnahme und Dokumentation der Anforderungen. Für jedes Applikationsmuster sind nun geeignete Datenstrukturen zu erarbeiten, welche die relevanten Daten abbilden. Ausgehend von der eigentlichen Tätigkeitsbeschreibung ist dabei vor allem die Frage nach dem fachbereichsübergreifenden Kommunikations- und Abstimmungsbedarf ausschlaggebend. Daraus ergibt sich die relevante Menge an Informationen,

die in den Applikationsmustern abgebildet werden muss, um den Datenaustausch durch die virtuelle Werkzeugmaschine unterstützen zu können. Für das Beispiel aus Abbildung 5-4 ist etwa der Austausch der CAD-Produktstruktur zwischen den drei Rollen erforderlich, um die Anforderungen und die davon abgeleiteten Funktionsbeschreibungen mit der Umsetzung durch die mechanischen Komponenten abgleichen zu können.

Als Grundlage für das Applikationsmuster *MCAD* dienen das Applikationsprotokoll AP 203 sowie die ISO-10303-21. Die Modellelemente zur Darstellung von Baugruppen und Bauteilen (Assemblies und Parts) werden in der AP 203 als so genannte *Products* abgebildet. In Abbildung 5-5 a ist ein vereinfachtes YAT-Modell für die Assembly-Definition, wie sie im *STEP physical exchange file* der ISO 10303-21 definiert wird, zu sehen. Eine Assembly-Definition wird durch die Entität *NEXT_ASSEMBLY_USAGE_OCCURENCE* eingeleitet.



a) Produktstruktur nach ISO 10303-21 b) Instanziierte Produktstruktur

Abbildung 5-5: Darstellung der Datenstrukturen in YAT

Die Attribute *id* und *name* sind für die eindeutige Kennzeichnung sowie für die Bezeichnung des Assemblies zuständig. Darüber hinaus werden zwei Attribute vom Entitäts-Typ *Product_Definition* angegeben, die letztendlich auf die *Product-Entitäten* verweisen, aus denen sich das Assembly-Konstrukt zusammensetzt. Die Entität *Product_Definition_Formation_With_Specified_Source* gibt dabei die Version der Products an. Die Entität *Product* selbst wird durch die drei Parameter *id*, *name* und *description* spezifiziert [ISO 2004]. Eine mögliche Instanz des YAT-Modells aus Abbildung 5-5 a ist in Abbildung 5-5 b zu sehen. Das beschriebene Beispiel zeigt die Vorgehensweise, um die Produktstruktur aus einer Datei im ISO 10303-21-Format auslesen zu können. Sollen zusätzliche Informationen für die virtuelle Werkzeugmaschine aufbereitet werden, ist eine weitere Detaillierung des YAT-Modells aus Abbildung 5-5 erforderlich.

Die Abbildung als YAT-Modell bietet die Möglichkeit, die Datenstrukturen der Softwarewerkzeuge übersichtlich in Baumdarstellung visualisieren zu können. Insbesondere dann, wenn die einzelnen Datenformate auf ähnliche Strukturen hin untersucht werden, bietet diese graphische Veranschaulichung Vorteile. YAT ist somit als ein Hilfsmittel zu begreifen, um die verschiedenen Datenstrukturen übersichtlich und transparent visualisieren zu können. Nach Fertigstellung der Arbeiten liegen mehrere Applikationsmuster als UML-Pakete sowie jeweils eine Beschreibung der einzelnen Datenformate in Form von YAT-Modellen vor.

Einordnung der Datenstrukturen und Modellelemente in die Ontologie

Die im vorhergehenden Schritt erstellten YAT-Modelle beschreiben die relevanten Daten der jeweiligen Applikationsmuster. Diese Modelle orientieren sich stark an den jeweiligen fachbereichsspezifischen Strukturierungsmitteln und Bezeichnungen. Für eine Integration in das Metamodell müssen die YAT-Modelle deshalb weiter abstrahiert werden. Das Ziel ist es, aus den einzelnen Graphen gemeinsame Strukturen und Modellelemente abzuleiten, um sie bei der Beschreibung der Applikationsmuster wieder verwenden zu können. In Abbildung 5-6 a ist ein YAT-Modell *YMCAD* zu sehen, das ebenfalls den Aufbau des Produktstrukturbaums *B-Einheit* aus Abbildung 5-5 b beschreibt. Die beiden YAT-Modelle *Assembly* und *YMACD* aus Abbildung 5-5 a erlauben es demnach, einen Produktbaum wie in Abbildung 5-5 b aufzubauen. Beide Graphen beschreiben genau die relevanten Daten, die in dem Szenario aus Abbildung 5-4 ausgetauscht werden sollen. Dabei ist der Graph aus Abbildung 5-6 a in seinem Aufbau einfacher und allgemeiner gehalten und bietet sich deshalb als Grundlage für die Definition des Applikationsmusters MCAD an. Das YAT-Modell *Produktstruktur B-Einheit* ist eine Instanz des Modells *YMCAD*.

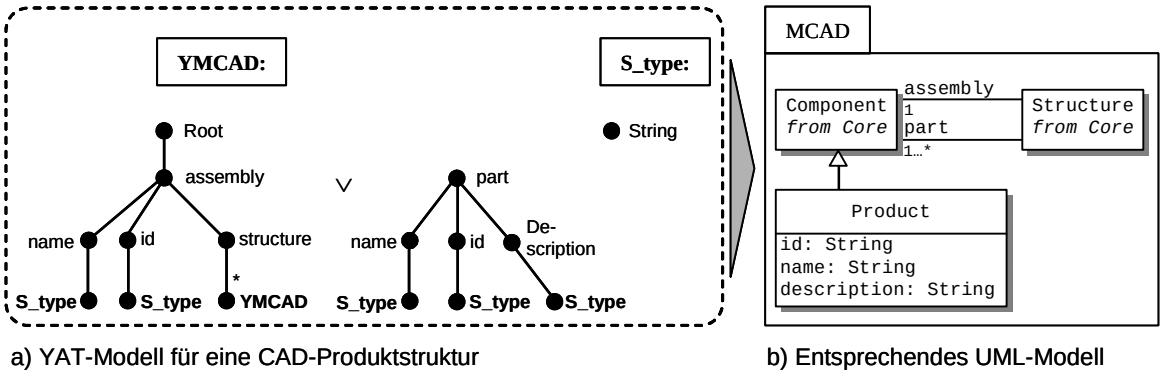


Abbildung 5-6: Ableiten der Klassendiagramme der Applikationsmuster aus den YAT-Modellen

In Abbildung 5-6 b findet sich das entsprechende UML-Klassendiagramm. Es ist von großer Bedeutung, dass die Bezeichnung der Klassen weitestgehend den Begriffen aus dem YAT-Modell entspricht, damit die Herleitung des Klassendiagramms nachvollziehbar bleibt. Des Weiteren werden die YAT-Graphen bei der Konsolidierung des Metamodells eingesetzt und tragen zur Definition der Daten-Anfragesprache bei. Es ist deshalb darauf zu achten, dass sich beide Darstellungen in ihren wesentlichen Strukturen und Begrifflichkeiten entsprechen. Bei der Definition der Applikationsmuster soll bewusst auf bereits vorhandene Klassen zurückgegriffen werden. Dadurch ist es möglich, gemeinsame Modellelemente und Abhängigkeiten zwischen den Fachbereichen zu explizieren und das Metamodell insgesamt zu vereinfachen. Im Beispiel aus Abbildung 5-6 b ist die Klasse *Component* verwendet worden, ohne dass dieser Begriff bei der Beschreibung der Datenstruktur durch das YAT-Modell verwendet wurde. Gleichwohl wurde diese Klasse bei der ersten, groben Begriffsbestimmung und Einordnung der Fachbereiche (siehe Abbildung 5-3) identifiziert und in das UML-Paket *Core* eingefügt. Für die Definition des Applikationsmusters *MCAD* kann und soll auf diese Klasse zurückgegriffen werden. Wie im Beispiel zu sehen, lässt sich im Metamodell mit dem Applikationsmuster *MCAD* eine Produktstruktur durch die Klasse *Structure* beschreiben, welche eine

Menge von *Component*-Klassen miteinander in Beziehung setzt. Für die Beschreibung von Produktstrukturen, die auf der ISO 10303-203 [ISO 10303-203 2004] basieren, wird die Klasse *Product* dem Applikationsmuster hinzugefügt und von der Klasse *Component* abgeleitet. Somit können Assembly-Part-Strukturen des AP 203 abgebildet werden. Auf dieselbe Art und Weise werden die UML-Klassendiagramme der weiteren Applikationsmuster erarbeitet. Wenn möglich, sollen dabei die allgemein gültigen Klassen aus dem *Core*-Paket verwendet werden. Dadurch können die einzelnen Applikationsmuster besser integriert werden.

Definition der Kern-Ontologie und der Applikationsmuster

Im Rahmen dieses Prozessschrittes werden die erarbeiteten Applikationsmuster in ihrem Aufbau einander angeglichen. Allgemein gültige Begriffe und Strukturen werden im *Core*-Paket zusammengefasst und können durch entsprechende Vererbungsbeziehungen in den einzelnen Applikationsmustern wieder verwendet werden. Diese Aufgabe wird zum Teil in den vorhergehenden Schritten intuitiv umgesetzt und nun gezielt zur Konsolidierung des gesamten Metamodells forciert. Das Klassendiagramm aus Abbildung 5-6 b zur Darstellung hierarchischer Strukturen kann beispielsweise auch für die Gliederung von Funktionen in Funktionsgruppen und Funktionseinheiten verwendet werden. Des Weiteren ist die Spezifikation von Maschinenfunktionen von derart grundlegender, allgemein gültiger Bedeutung, dass dieser Sachverhalt im *Core*-Paket definiert wird. In den Applikationsmustern wird die konkrete Ausprägung der Funktionsspezifikation entsprechend den Anforderungen des Fachbereichs detailliert ausgeführt. Das *Core*-Paket implementiert demnach die Kern-Ontologie O^c aus Definition 4-7 und wird im weiteren Verlauf der Arbeit mit dem Begriff *Metamodell-Kern* bezeichnet. In Abbildung 5-7 sind die Pakete des Metamodells für das Beispielszenario aus Abbildung 5-4 dargestellt.

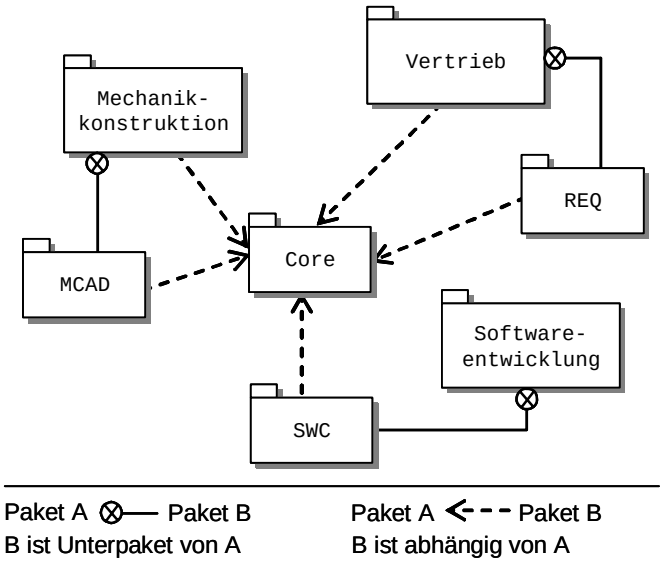


Abbildung 5-7: Aufbau des Metamodells durch Pakete

Die Applikationsmuster sind als Unterpakete den einzelnen Fachbereichen zugeordnet und verwenden Klassen aus dem *Core*-Paket zur Definition der fachbereichsspezifischen Modellelemente und Datenstrukturen.

5.2.3 Zusammenfassung

Die Definition des Metamodells hängt von den im Unternehmen eingesetzten Modellierungstechniken und Softwarewerkzeugen ab. Ein einfacher Austausch des Metamodells zwischen verschiedenen Unternehmen ist daher nur bedingt möglich. Der Metamodell-Kern selbst stellt eine Ausnahme dar, da er abstrahierte Modell- und Datenelemente beschreibt, die für den Bereich der Werkzeugmaschinenentwicklung allgemein gültig sind. Die beschriebene Vorgehensweise umfasst den gesamten Ablauf zum Aufbau des Metamodells. Für die Integration neuer Applikationsmuster in ein bereits bestehendes Metamodell sind vorwiegend die Schritte 3 bis 5 iterativ zu bearbeiten. Ist mit der Einführung eines neuen Softwarewerkzeuges in die virtuelle Werkzeugmaschine eine Änderung der Abläufe verbunden, ist gegebenenfalls eine erneute Analyse derselben durchzuführen.

Da der Aufbau des Metamodells eng an die Umsetzung der Entwicklungsprozesse sowie die eingesetzten Methoden und Werkzeuge im Unternehmen geknüpft ist, ist eine allgemein gültige Beschreibung des Metamodells nicht möglich. Aus diesem Grund wird im folgenden Abschnitt eine Modellierungstechnik für eine systemtechnisch orientierte Modellierung von Werkzeugmaschinen vorgestellt. Die Beschreibung des Metamodells im Anschluss bezieht sich auf die dabei verwendeten Modellkonzepte.

5.3 Systemmodellierung mit der virtuellen Werkzeugmaschine

Im weiteren Umfeld der vorliegenden Arbeit, d. h. am Institut für Werkzeugmaschinen und Betriebswissenschaften der TU München (*iwb*), wurden mehrere Modellkonzepte entwickelt, die bestimmte Aspekte der Werkzeugmaschine mit geeigneten Beschreibungsmitteln abbilden. So wurde im Projekt *MECHASOFT* ein Konzept zur Beschreibung der Maschinenfunktionen und zur Modellierung der Störungsbehandlung [REINHART ET AL. 2002C] entworfen. Ein Funktionsmodell der Maschine fungiert hierbei als Kommunikations- und Spezifikationsgrundlage für alle weiteren Konstruktionsschritte. Ziel des Projektes war es, das klassische Abteilungsdenken der Unternehmen aufzubrechen und bereits in den frühen Phasen der Entwicklung ein einheitliches, interdisziplinäres Modellkonzept einzuführen [REINHART ET AL. 2001B].

Die Arbeiten in MECHASOFT werden erweitert durch das von der Forschungsvereinigung Werkzeugmaschinen und Fertigungstechnik e.V. (FWF) geförderte Projekt *Simulation von Maschinenabläufen an virtuellen Werkzeugmaschinen* [ZAEH & POERNBACHER 2005]. Aufbauend auf einem systemtheoretischen Ansatz werden die Hardware- und Softwarekomponenten der Werkzeugmaschine hierarchisch in Subsysteme zergliedert. Diesen kann ein Verhalten zugewiesen werden, wobei die Kommunikation zwischen den Subsystemen über definierte Schnittstellen erfolgt. Das so erstellte Modell kann zur PLC-Codegenerierung und zur anschließenden Validierung der Steuerungssoftware eingesetzt werden.

Im Rahmen dieser Arbeit werden die Modellierungstechniken und Vorgehensweisen aus den beiden Projekten exemplarisch als Referenzmethode für die Beschreibung von Maschinenfunktionen verwendet und in einem einheitlichen Systemmodellierungsansatz zusammengeführt.

In Abbildung 5-8 ist das gesamtheitliche Vorgehen bei der modellgestützten Entwicklung von Werkzeugmaschinen zu sehen.

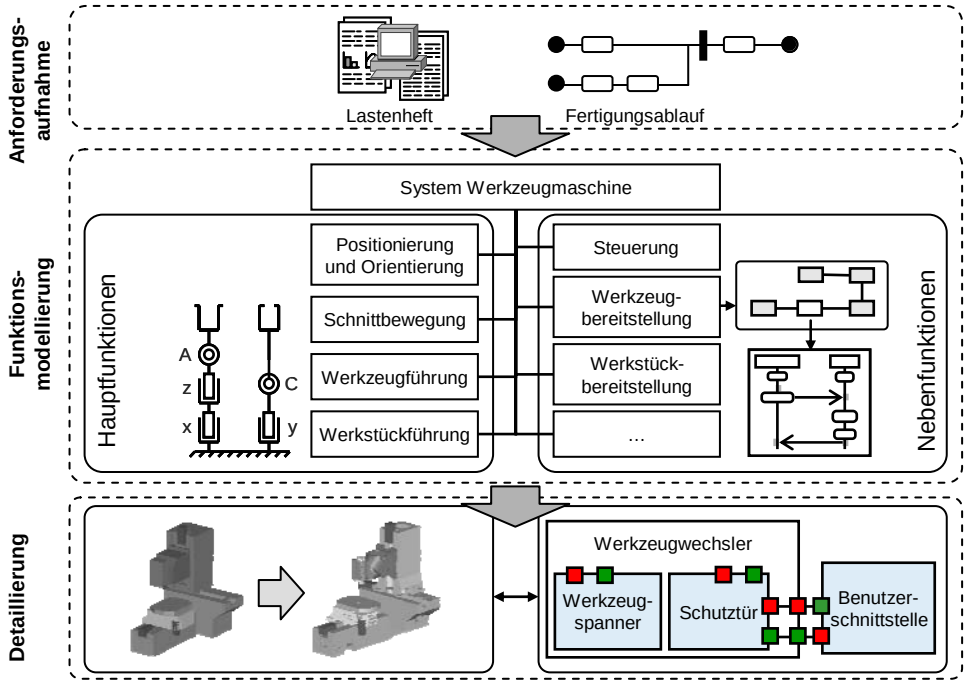


Abbildung 5-8: Modellgetriebene Entwicklung mit der virtuellen Werkzeugmaschine

In den folgenden Abschnitten werden die einzelnen Schritte des abgebildeten Vorgehens beschrieben. Der Schwerpunkt liegt dabei auf der Erläuterung der eingesetzten Modellelemente und deren fachbereichsübergreifenden Abhängigkeiten. Eine ausführliche Beschreibung der Methodik ist in [ANTON ET AL. 2002] und [REINHART ET AL. 2002C] zu finden.

5.3.1 Anforderungsaufnahme

Der Werkzeugmaschinenhersteller erhält vom Kunden das Lastenheft mit detaillierten Vorgaben über zu verwendende Baugruppen wie beispielsweise Antriebe, Zylinder oder Steuerungskomponenten. Im Sondermaschinenbau ist zudem häufig ein vom Kunden geforderter technologischer Prozess der Ausgangspunkt der Entwicklungsarbeiten. Neben den allgemeinen Vorgaben hinsichtlich der Gestellstruktur und der Taktzeiten spielen auch Arbeits- und Schnittfolgen beim Zerspanprozess eine große Rolle [REINHART ET AL. 2002C]. Um den einzelnen Fachbereichen die für sie relevanten Informationen direkt zugänglich zu machen, werden die im Lastenheft dokumentierten Informationen mit entsprechenden Metadaten annotiert. Diese Metadaten beschreiben den Kontext einer Anforderungs- oder Merkmalbeschreibung und ermöglichen die Einordnung in definierte Anforderungsklassen sowie die Zuordnung zu bestimmten Fachbereichen. In Abbildung 5-9 a ist beispielhaft ein Lastenheft des Kunden zu sehen, das vom Werkzeugmaschinenhersteller zur weiteren Verarbeitung bereits annotiert wurde.

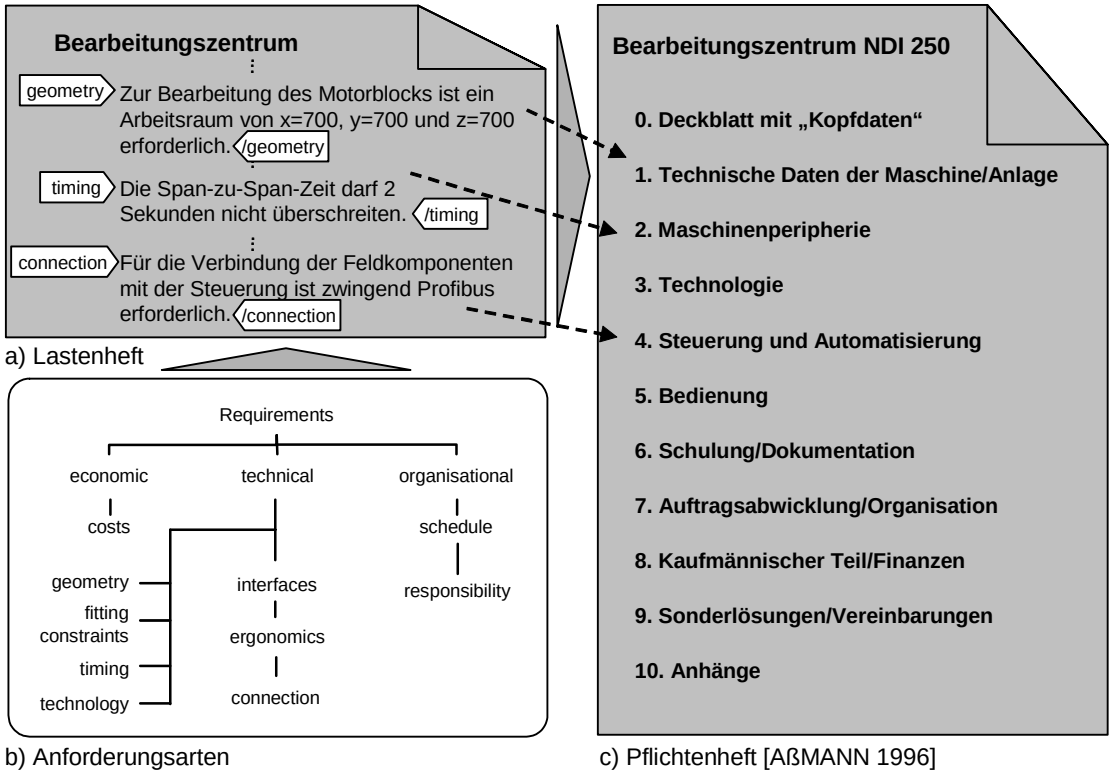


Abbildung 5-9: Anforderungsaufnahme und –strukturierung sowie zugehöriges Pflichtenheft (exemplarisch)

Das Lastenheft wird häufig mit einem herkömmlichen Textverarbeitungssystem erstellt. Die Informationen, wie beispielsweise die Angaben über den erforderlichen Arbeitsraum, sind semistrukturiert im Fließtext enthalten. Der Dokumentabschnitt im Lastenheft mit den geforderten Achslängen wird mit dem Metadatum *geometry* ausgezeichnet, um darzustellen, dass hier geometrische Informationen enthalten sind. Dementsprechend wird der Textabschnitt mit den Vorgaben bezüglich der Span-zu-Span-Zeit mit einem *timing*-Element annotiert. Die Metadaten bezeichnen verschiedene Anforderungstypen, welche unternehmensweit festgelegt sind. In Abbildung 5-9 b ist eine mögliche Klassifizierung von Anforderungsarten nach [EHRLENSPIEL 1995] dargestellt. Durch die Annotation ist es möglich, die einzelnen Dokumentabschnitte des Lastenheftes für die internen Abläufe des Werkzeugmaschinenherstellers aufzubereiten und in ein Pflichtenheft für die einzelnen Fachbereiche zu überführen. In Abbildung 5-9 c ist ein Pflichtenheft nach dem Vorschlag von [AßMANN 1996] zu sehen. Die Zuordnung der einzelnen Textabschnitte des Lastenhefts zu den einzelnen Kapiteln des Pflichtenhefts kann einfach über die Annotationen erfolgen. Das Hinzufügen der hierfür erforderlichen Metadaten zum Lastenheft wird durch die Definition entsprechender Formatvorlagen umgesetzt. Diese implementieren die in Abbildung 5-9 b aufgeführten Anforderungsarten. Die Umsetzung der Annotationen erfolgt beispielsweise durch die Definition entsprechender Formatvorlagen in herkömmlichen Textverarbeitungssystemen.

5.3.2 Funktionsmodellierung

Aus dem Pflichtenheft werden nun die für die Umsetzung der Maschineneigenschaften erforderlichen Maschinenfunktionen abgeleitet. Diese lassen sich prinzipiell in Haupt- und Nebenfunktionen untergliedern (siehe Abbildung 5-8). Bei der Spezifikation der Hauptfunktionen liegt der Fokus auf der Umsetzung des geforderten technologischen Prozesses und der Aufnahme des durch den Zerspanprozess induzierten Kraftflusses. Insbesondere erfolgt hier die Festlegung der Anzahl und der Lage der Achsen, wovon sich wiederum die prinzipielle Kinematik und das Konzept der Maschine ableiten. Als Beschreibungsmittel kommen vorwiegend 3D-CAD-Modelle sowie 2D-Skizzen der Schnittfolgen, die den Bearbeitungsprozess anhand einer Werkstück-Zeichnung verdeutlichen, zum Einsatz. Daraus leitet sich der technologische Zerspanprozess ab. Bei den CAD-Modellen kann es sich um Konzepte aus bereits abgeschlossenen Projekten oder um völlige Neukonstruktionen handeln.

Für die Definition der Nebenfunktionen wird der Ansatz der Szenarioanalyse verfolgt, um die Maschinenabläufe und die Interaktionen einzelner Komponenten untereinander herausarbeiten zu können. Die aufgeführten Anforderungen werden hierfür als Use Cases in ein UML-Case-Tool importiert und dort in Aktivitäts- und Sequenzdiagrammen weiter ausgeführt. In Abbildung 5-10 ist ein Aktivitätsdiagramm zu sehen, welches den Ablauf der Nebenfunktion *Werkzeugbereitstellung* näher beschreibt.

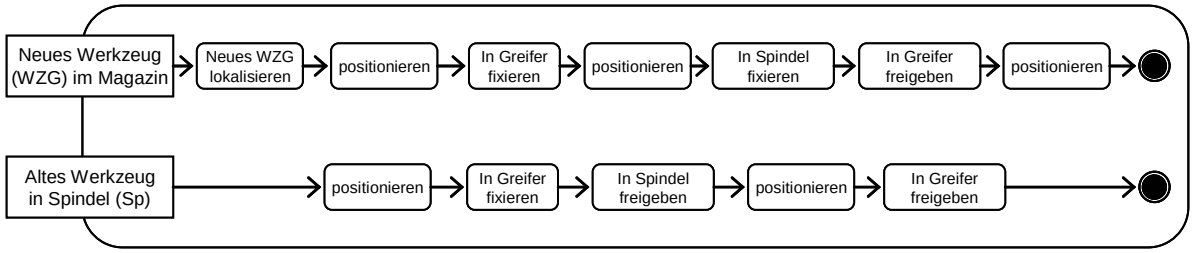


Abbildung 5-10: Funktion „Werkzeugbereitstellung“ als an die UML angelehntes Aktivitätsdiagramm [REINHART ET AL. 2001A]

Die Funktion wird als Aktivität durch ein an den Ecken abgerundetes Rechteck dargestellt. Der von einer Aktivität beschriebene Ablauf wird durch so genannte Aktionen beschrieben. Aktionen können als Teilfunktionen der Aktivität verstanden werden. Die Aktivitäten sind parametrierbar. Im vorliegenden Beispiel werden das auszuwechselnde Werkzeug in der Spindel sowie das Werkzeug im Magazin als Eingabe-Parameter übergeben. Der Ablauf selbst beschreibt den Wechselvorgang und gibt Aufschlüsse über die daran beteiligten Baugruppen. Aktionen können wiederum neue Aktivitäten aufrufen, so dass ein hierarchisches Funktionsmodell der Maschinenfunktionen entsteht. Dieser Arbeitsschritt kann iterativ wiederholt werden, bis geeignete Lösungselemente für die einzelnen Aktionen festgelegt werden können. Dieses Diagramm wird von einem interdisziplinären Team erstellt und ermöglicht es, ein gemeinsames Problemverständnis zu entwickeln.

Ausgehend von den Ablaufbeschreibungen im Aktivitätsdiagramm können nun die an der Umsetzung der Funktion beteiligten Baugruppen und Bauteile identifiziert werden. Dies erfolgt anhand von UML-Sequenzdiagrammen. Sie ermöglichen die Visualisierung der Interak-

tionen zwischen den Komponenten auf der Grundlage der zuvor definierten Maschinenabläufe. Das Sequenzdiagramm dient insbesondere dem Softwareentwickler als Ausgangsgrundlage für die Programmierung der Steuerungssoftware. Es lassen sich jedoch auch Informationen für die Planung der elektrischen Betriebsmittel ableiten. Bereits im Aktivitätsdiagramm ist es möglich, den Aktionen Signale und Ereignisse zuzuordnen, welche zur Identifikation dezidierter Maschinenzustände herangezogen werden. Diese Signale und Ereignisse werden im Sequenzdiagramm als Nachrichten beschrieben, die beispielsweise von einem Sensor im Maschinenfeld zur SPS-Steuerung gesendet werden. Abbildung 5-11 zeigt ein Sequenzdiagramm für den in Abbildung 5-10 dargestellten Ablauf.

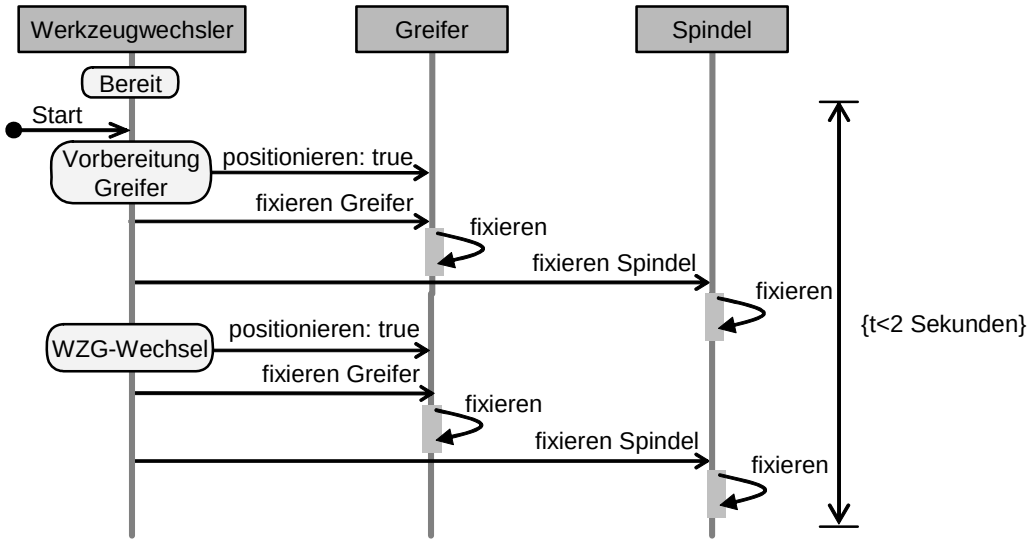


Abbildung 5-11: Interaktionen zwischen den Baugruppen und Bauteilen

Sequenzdiagramme ordnen verhaltensspezifische Aspekte einer Werkzeugmaschine in einen strukturellen Kontext ein. Die Interaktionen zwischen den einzelnen Baugruppen und Bauteilen werden in zwei Dimensionen beschrieben. In einer horizontalen Ebene von links nach rechts werden die an der Interaktion beteiligten Komponenten bzw. Kommunikationspartner eingezeichnet. Auf einer senkrecht von oben nach unten verlaufenden Zeitachse sind die Nachrichten bzw. Signale durch Pfeile gekennzeichnet. Die Aktionen aus dem Aktivitätsdiagramm werden als Balken auf den Lebenslinien eingetragen. Vorbedingungen für Interaktionen werden durch so genannte *Zustandsinvarianten* definiert. Zustandsinvarianten werden durch Rechtecke mit abgerundeten Ecken dargestellt und auf den Lebenslinien der Komponenten eingezeichnet. Eine Kommunikation kann nur dann erfolgen, wenn sich die Sender- bzw. Empfängerkomponente in dem Zustand befindet, der von einer Zustandsinvariante beschrieben wird. Zusätzlich ist die Angabe weiterer Randbedingungen möglich, die sich beispielsweise direkt aus der Anforderungsspezifikation ergeben. So können zeitliche Vorgaben des Kunden in das Sequenzdiagramm übernommen und auf verschiedene Teilfunktionen verteilt werden. Im Beispiel ist die Zeitvorgabe des Kunden bezüglich der Span-zu-Span-Zeit im Sequenzdiagramm abgebildet. Das Sequenzdiagramm wird zunächst vom Mechanikkonstrukteur angefertigt und mit der Elektrokonstruktion diskutiert. Anschließend wird das Sequenzdiagramm gemeinsam schrittweise ausformuliert und konkretisiert. Der Formulierung der Zustandsinvarianten kommt dabei ein besonderer Stellenwert zu. Durch sie können die Bedingungen festgelegt werden, die für die Ansteuerung der Feldkomponenten einzuhalten sind.

Das Sequenzdiagramm ist somit ein Hilfsmittel, um die zur Abbildung der Maschinenzustände notwendigen Signalabbilder methodisch zu erfassen. Weitere Ausführungen hierzu sind [ZÄH & POERNBACHER 2005] zu entnehmen.

Um ein besseres interdisziplinäres Verständnis zu gewährleisten, wird parallel zum Sequenzdiagramm eine *Technologieskizze* erstellt. Dabei handelt es sich um ein 2D-Modell, das genau die Komponenten darstellt, die zur Umsetzung des Szenarios aus dem Sequenzdiagramm erforderlich sind. In Abbildung 5-12 ist die Technologieskizze für die Baugruppe *Werkzeugwechsler* zu sehen. Die mechanische Struktur wird lediglich skizziert, um die Orientierung zu erleichtern.

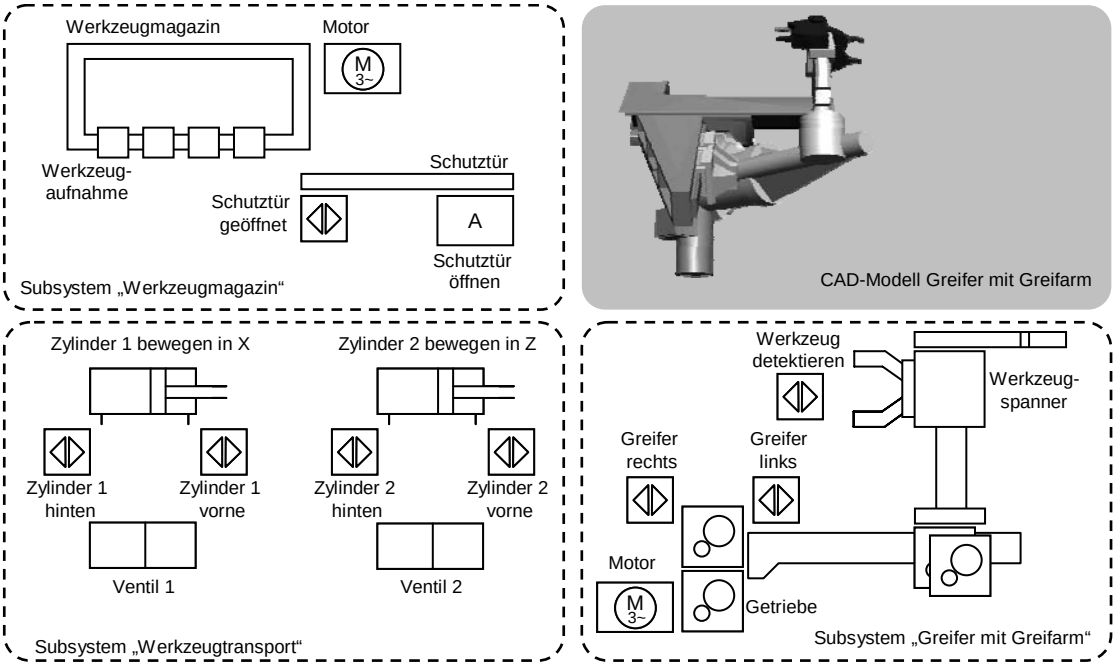


Abbildung 5-12: Technologieschema zur Darstellung von Wirkprinzipien und Lösungskonzepten

Für die Abbildung der Aktoren, der Sensoren und der Stellglieder werden die aus den Normen *DIN 40900, DIN ISO 1219* und *DIN 30600* bekannten Symbole verwendet. Die wesentliche Aufgabe der Technologieskizze besteht darin, das Funktionsprinzip einzelner Baugruppen transparent abzubilden und dabei die erforderlichen elektrischen Betriebsmittel zu berücksichtigen. Sequenzdiagramm und Technologieskizze bauen aufeinander auf und bilden zusammen das *Technologieschema* [REINHART ET AL. 2002A]. Auf der Basis des Technologieschemas können das 3D-CAD-Modell aufgebaut sowie die für die Steuerung der Maschinenabläufe notwendigen Signale abgeleitet werden. Im folgenden Abschnitt werden die hierfür eingesetzten Beschreibungsmittel vorgestellt.

5.3.3 Detaillierung

Nach der Auswahl des Lösungskonzepts werden die Technologieschemata für die Darstellung der Nebenfunktionen sowie das 3D-CAD-Modell zur Beschreibung der Hauptfunktionen zusammengeführt. Aus den zuvor erstellten Modellen und Diagrammen kann der Softwareentwickler nun die Struktur der Steuerungssoftware entwerfen. Als Beschreibungsmittel werden hierfür UML-Komponentendiagramme [JECKLE ET AL. 2004] eingesetzt (siehe Abbildung 5-13).

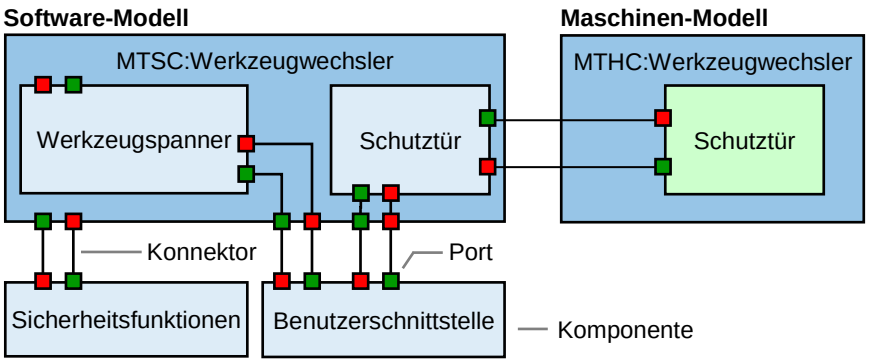


Abbildung 5-13: Komponentendiagramm zur Spezifikation der Softwarestruktur [ZÄH & POERNBACHER 2005]

Die einzelnen Subsysteme der Werkzeugmaschine werden dabei durch das Modellelement *Component* repräsentiert. Komponenten können verschachtelt werden, wodurch der Aufbau hierarchischer Strukturen möglich ist. Im Beispiel aus Abbildung 5-13 besteht die Komponente *Werkzeugwechsler* aus dem *Werkzeugspanner* und der *Schutztür*. Jedem Component-Element kann eine beliebige Anzahl an Schnittstellen zugewiesen werden, durch die Nachrichten mit weiteren Systemkomponenten ausgetauscht werden können. Die Schnittstellen werden im Komponentendiagramm durch *Ports* dargestellt. Sie beschreiben die Informationen, die von einer *Component* entweder zur Verfügung gestellt oder benötigt werden. Die Verbindung mehrerer Komponenten durch *Ports* wird durch die Konnektoren visualisiert. Konnektoren bilden den Informationsfluss im System ab. Im Beispiel (siehe Abbildung 5-13) ist die Verbindung des Softwaremodells der Schutztür mit dem entsprechenden Maschinenmodell zu sehen. Der Vorteil dieser Trennung liegt unter anderem darin, dass das diskrete und kontinuierliche Systemverhalten der Komponenten jeweils mit geeigneten Beschreibungsmitteln abgebildet werden kann. Die SPS beispielsweise besitzt ein diskretes Systemverhalten, das sich durch die zyklische Arbeitsweise ergibt. Deshalb kann zur Verhaltensbeschreibung ein Zustandsgraph verwendet werden (siehe Abbildung 5-14).

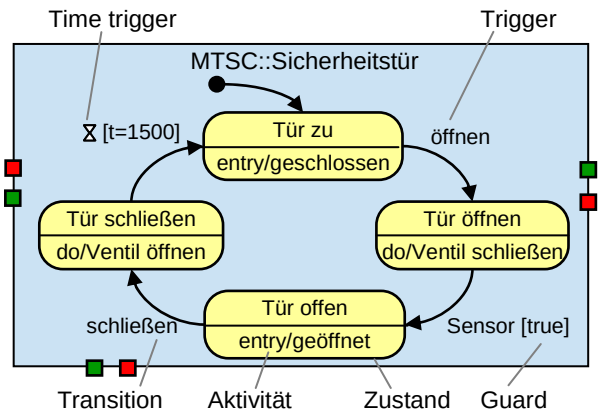


Abbildung 5-14: Zustandsgraph für die Komponente Schutztür im Softwaremodell [ZÄH & POERNBACHER 2005]

Zustandsgraphen bestehen aus einer endlichen Menge von Zuständen und Transitionen, wobei zu jedem Zeitpunkt genau ein Zustand aktiv ist. Zustandsübergänge werden durch Änderungen der Signalabbilder der Maschine ausgelöst. Die Trigger legen dabei fest, welche Signale an einer Transition anliegen müssen, um einen bestimmten Zustandsübergang auszulösen. Darüber hinaus existieren spezielle Ausprägungen von Triggern wie beispielsweise *Time-Trigger*. Sie veranlassen einen Zustandsübergang zu einer bestimmten Zeit bzw. nach einer vorher definierten Zeitspanne. Des Weiteren können zusätzliche Ausführungsbedingungen in Form so genannter *Guards* angegeben werden. Diese Bedingungen müssen erfüllt sein, bevor ein Zustandsübergang möglich ist.

Für eine gesamtheitliche, modellgestützte Beschreibung der Werkzeugmaschine ist auch eine hinreichende Abbildung des Verhaltens der Hardwarekomponenten erforderlich. Hier sind vor allem das Zeitverhalten, das logische Verhalten sowie das Störungsverhalten von Bedeutung. Deren Abbildung erfolgt durch eine blockorientierte Modellierung mit Beschreibungsmitteln der Sprache Modelica (siehe Abbildung 5-15).

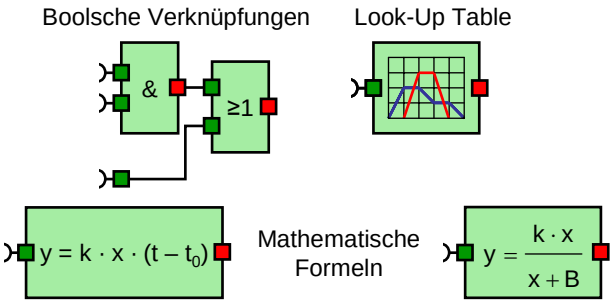


Abbildung 5-15: Blockorientierte Modellierung des Maschinenverhaltens [ZÄH & POERNBACHER 2005]

Zum Einsatz kommen dabei unter anderem einfache Zeitglieder, mathematische Funktionen sowie *Look-Up Tables*, in denen die Ausgangswerte in Abhängigkeit eines Eingangswertes bestimmt werden. In [ZÄH ET AL. 2003] und [EHRENSTRASSER ET AL. 2003] wird diese Form der Modellierung für die Hardware-in-the-Loop-Simulation eingesetzt. Hierbei wird das zu-

vor erläuterte Funktionsmodell an eine dreidimensionale Visualisierung der Werkzeugmaschine angekoppelt, wodurch die Spezifikation und Verifikation der entwickelten Steuerungsfunktionen erleichtert wird.

Auf der Grundlage der zuvor erarbeiteten Anforderungsspezifikation und der gemeinsam mit weiteren Fachbereichen erstellten Funktionsbeschreibung detailliert die Mechanikkonstruktion das CAD-Modell der Werkzeugmaschine. Dieser Fachbereich bestimmt im Wesentlichen die Produktstruktur, ist für die Umsetzung der Funktionsspezifikationen zuständig und führt gegebenenfalls Schwachstellenanalysen auf Basis der MKS bzw. FEM durch.

Der Aufbau des gesamten Funktionsmodells erfolgt interdisziplinär mit verschiedenen, fachbereichsspezifischen Softwarewerkzeugen. Damit die virtuelle Werkzeugmaschine den Anwender bei der Erstellung und Verifikation unterstützen kann, ist es erforderlich, dass die zentralen Beschreibungsmittel des Funktionsmodells aufeinander abgestimmt werden. Diese Aufgabe übernimmt das Metamodell, das im folgenden Abschnitt vorgestellt wird.

5.4 Aufbau des Metamodells

Für die Realisierung des Metadaten-Management-Systems ist das in Abschnitt 4.5 erläuterte formale Modell in implementierungsnahe Datenstrukturen zu überführen. Die in Definition ***4-7*** aufgeführten Elemente werden in einem MOF-kompatiblen Metamodell (siehe Abschnitt 2.3) abgebildet, wobei für die Visualisierung der fachlichen Konzepte die Notation der UML verwendet wird. In Abbildung 5-16 ist die entsprechende Überführungsmatrix dargestellt. Dabei gelten die folgenden Abbildungsvorschriften:

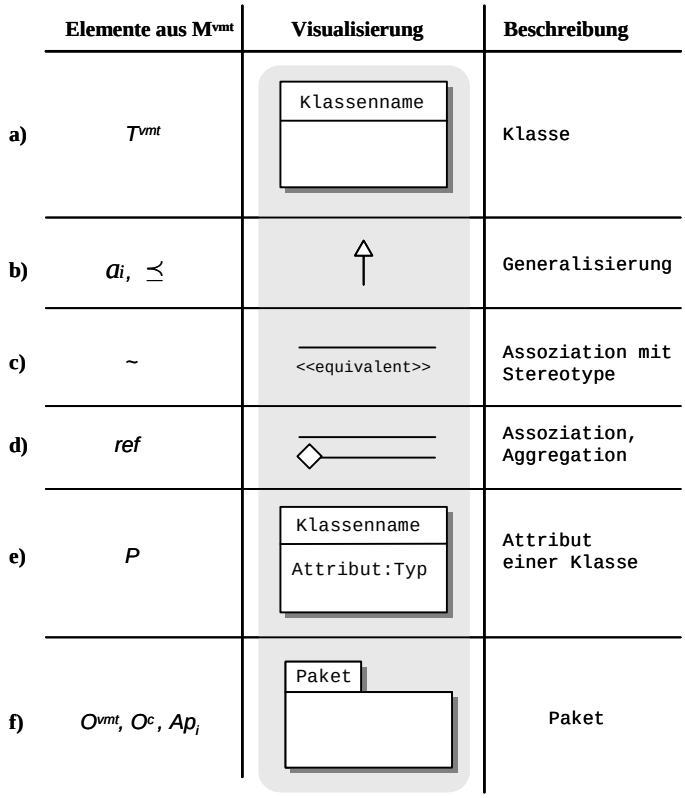


Abbildung 5-16: Abbildung der Elemente aus Definition 4-7 auf Modellkonstrukte der UML

Die in der Terminologie $T^{vmt} = T^c \cup T_i^{ap}$ zusammengefassten Begriffe der einzelnen Applikationsmuster T_i^{ap} sowie der Terminologie T^c des Kerns werden durch UML-Klassen visualisiert. Der Klassenname gibt dabei die Begriffsbezeichnung wieder (siehe Abbildung 5-16 a).

Zwischen den Klassen der UML können Beziehungen definiert werden. Diese Möglichkeit wird genutzt, um die Relationen des Mediatormodells aus Definition 4-7 abzubilden. Die Artikulationsbeziehungen a_i sowie die Zuordnungsrelationen $\preceq_i^{ap}$ und $\preceq^c$ werden anhand von Generalisierungs-Pfeilen abgebildet. Diese Relationen definieren eine partielle Ordnung auf der Begriffsmenge T^{vmt} und gliedern somit einzelne Begriffe in zusammengehörige Begriffswelten. Die Semantik der Generalisierung in der UML lässt eine Abbildung derartiger Strukturen zu (siehe Abbildung 5-16 b).

Für die Darstellung der Äquivalenzrelation ~ werden UML-Assoziationen mit dem Stereotype <<*equivalent*>> verwendet (siehe Abbildung 5-16 c). Allgemeine Referenzen zwischen verschiedenen Begriffen werden durch Assoziationen ohne nähere Definition bzw. Aggregationsbeziehungen modelliert (siehe Abbildung 5-16 d).

Begrifflichkeiten können durch Attribute weiter konkretisiert werden. Die Attribute P werden einfach als Klassenattribute der UML-Klassendefinition hinzugefügt (siehe Abbildung 5-16 e).

Die Strukturierung und Gliederung des Mediatormodells in die einzelnen Ontologiedefinitionen und Applikationsmuster erfolgt mit den von der UML bereitgestellten *Paketen*. Diese sind im Wesentlichen als Container zu betrachten, in denen die entsprechenden Klassenbeschreibungen und deren Beziehungen untereinander abgelegt werden (siehe Abbildung 5-16 f).

Der Aufbau des Metamodells gliedert sich nach Definition 4-7 sowie nach Abbildung 5-7 in verschiedene Applikationsmuster und in einen allgemeinen, neutralen Metamodell-Kern. In Abbildung 5-17 ist dieser Sachverhalt dargestellt.

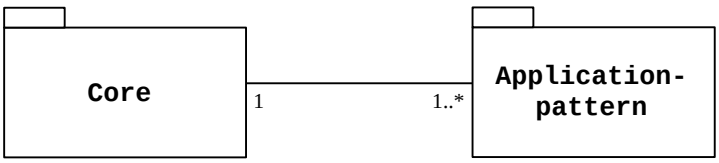


Abbildung 5-17: Aufbau des Machine-Tool-Metamodells

Wie in Abschnitt 5.2 erläutert, ergibt sich die Struktur des Metamodells entsprechend den in den einzelnen Unternehmen eingesetzten Modellierungstechniken und Softwarewerkzeugen. Der Metamodell-Kern nimmt hierbei die Rolle einer allgemeingültigen Basisstruktur ein und ist deshalb unabhängig von den unternehmensspezifischen Sachverhalten. Für jedes Softwarewerkzeug, das in den Informationsverbund der virtuellen Werkzeugmaschine integriert werden soll, ist ein entsprechendes, so genanntes *Applikationsmuster* (Applicationpattern) zu definieren. In ihm werden softwarespezifische bzw. fachbereichsspezifische Sprachkonstrukte auf der Basis des Metamodell-Kerns definiert.

Die Applikationsmuster erlauben somit ein Zuschneiden des Metamodells auf die Bedürfnisse der einzelnen Unternehmen.

5.4.1 Untersuchte Modellierungssprachen und Datenformate

Für die Definition des Metamodell-Kerns ist die Analyse unterschiedlicher Modellierungstechniken notwendig, um zentrale Basiskonstrukte identifizieren zu können, auf die die Sprachdefinitionen in den Applikationsmustern zurückgeführt werden können. Ziel hierbei war es, einen möglichst breiten Überblick über die verschiedenen Modellkonzepte zu erlangen. Der Schwerpunkt lag vor allem in der Analyse

- der Hierarchisierungs- und Strukturierungskonzepte sowie Schnittstellenbeschreibung,
- der Beschreibungsmittel für die Verhaltensbeschreibung (kontinuierlich bzw. diskret),
- der Unterstützung zur funktionalen Systembeschreibung sowie
- der Anwendung innerhalb existierender Methoden und Vorgehensweisen.

Für die Analyse wurden die Ergebnisse aus [VDI/VDE 3681 2005] und [CHOUIKHA ET AL. 1998] um weitere elementare Modellierungssprachen und Normen erweitert. Das Ergebnis ist in Tabelle 5-2 dargestellt.

Modellierungstechnik \ Kriterium	Formale Basis			Verhaltensbeschreibung	Abbildung der Zeit	Struktur		Abbildung von Prozessen			Darstellung			Werkzeugunterstützung	Methode
	Formal	Semi-formal	Informal			Hierarchie	Komposition / Dekomposition	Parallel	Sequentiell	Nebenläufig	Textuell	Mathematisch-symbolisch	Graphisch		
Ablaufsprache (AS)	○	●	○	●	●	●	●	○	○	●	○	○	●	●	○
Anweisungsliste (AW)	○	●	○	●	●	●	○	○	○	○	●	○	○	●	○
Blockschaltbilder	●	○	○	●	○	●	●	○	○	○	○	●	●	○	○
Bondgraphen	●	○	○	●	○	●	●	○	○	○	○	○	●	●	○
Funktionsbaustein-Sprache (FBS)	○	●	○	●	●	●	●	○	○	○	○	○	●	●	○
Funktionsblöcke nach IEC/DIN EN 61499	○	●	○	●	●	●	●	○	○	●	○	○	●	●	○
Funktionsmodellierung nach Pahl/Beitz, Ehrlenspiel, Roth, Koller	○	○	●	●	◐	○	◐	○	○	○	●	○	●	◐	●
Funktionsplan (FUP)	○	●	○	●	●	○	●	○	○	○	○	○	●	●	○
Impuls- und Zeitdiagramm	○	○	●	●	●	○	○	●	●	○	○	○	●	●	○
Kontaktplan (KOP)	○	●	○	●	●	●	○	○	○	○	○	○	●	●	○
Message Sequence Chart (MSC)	○	●	○	●	○	○	○	○	○	○	○	○	●	●	○
Petrinetze	●	●	○	●	●	●	○	○	○	●	○	○	●	●	○
Prinziplösungsmodellierung nach Kallmeyer	○	●	○	●	●	●	●	○	○	●	○	○	●	●	●
Sequential Function Chart (SFC)	○	●	○	●	●	○	●	●	●	○	○	○	●	●	○
STEP - ISO 10303-11	○	●	○	○	○	●	●	○	○	○	●	○	●	●	○
Specification and Description Language (SDL)	●	○	○	●	●	●	●	●	○	○	○	○	●	●	●
Strukturierter Text (ST)	○	●	○	●	●	●	○	○	○	○	●	○	○	●	●

● weitgehend erfüllt ◐ partiell erfüllt ○ nicht erfüllt

Tabelle 5-2: Einordnungskriterien von Modellierungssprachen und Normen

Das in den folgenden Abschnitten vorgestellte Metamodell für die virtuelle Werkzeugmaschine bezieht sich auf das im Rahmen des Forschungsprojektes *MECHASOFT* [ANTON ET AL. 2002] erarbeitete Funktionsmodell für Werkzeugmaschinen und auf die im Projekt *Simulation von Maschinenabläufen an virtuellen Werkzeugmaschinen* [ZAEH & POERNBACHER 2005] vorgestellten Methoden zur Maschinenmodellierung und -simulation (siehe Abschnitt 5.3.3).

5.4.2 Das Core-Paket

Prinzipiell werden im Metamodell-Kern zwei Arten von Elementen unterschieden. Die so genannten *aktiven Sprachkonstrukte* stellen Elemente dar, die direkt in den jeweiligen fachbereichsspezifischen Modellinstanzen für die Abbildung von Maschineneigenschaften verwendet werden und für die XML-Adapter als eigenständiges Sprachelement interpretierbar sind. Als Beispiel seien hier die Modellkonstrukte *Assembly* bzw. *Part* aus der Mechanikkonstruktion oder der *Functionblock* nach IEC 61131-3 aus der Steuerungsentwicklung genannt. Im Gegensatz dazu sind *passive Sprachelemente* nicht Teil von Modellbeschreibungen. Sie sind notwendig, um aktive Sprachkonstrukte zu klassifizieren oder in Strukturen einzugliedern. Passive Sprachkonstrukte werden im Metamodell-Kern als abstrakte Klassen abgebildet. Dementsprechend sind deren Bezeichnungen kursiv dargestellt. In Abbildung 5-18 ist das UML-Klassendiagramm des Metamodell-Kerns zu sehen.

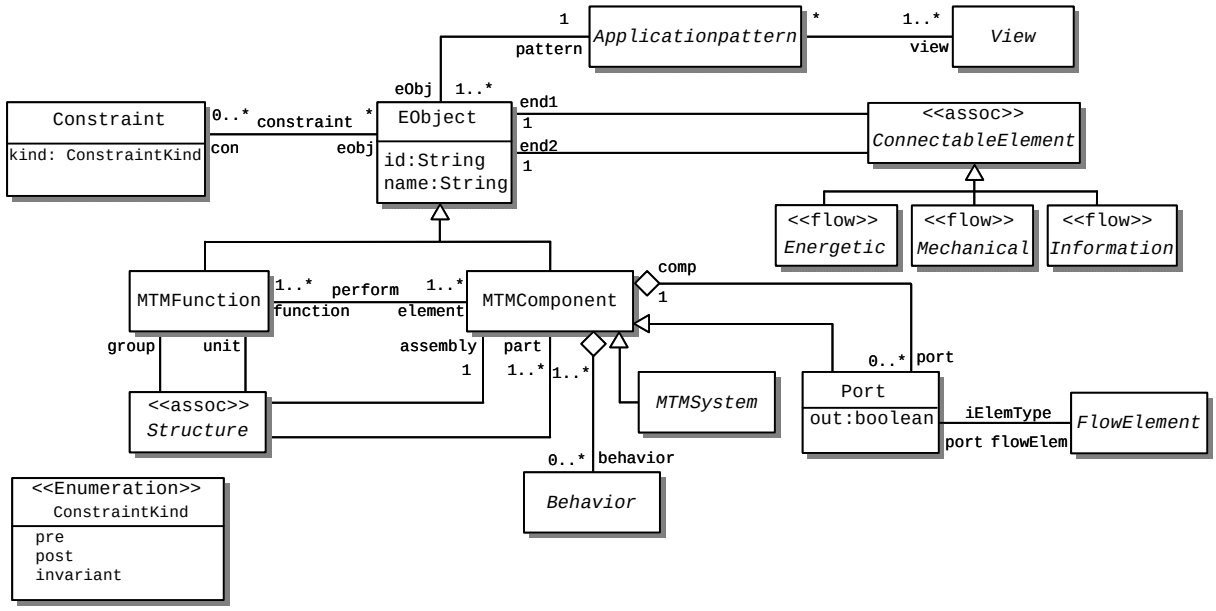


Abbildung 5-18: Klassendiagramm des Core-Pakets

Der im *Core*-Paket definierte Metamodell-Kern kann als Extrakt jener Sprachkonstrukte bezeichnet werden, die in den einzelnen Fachabteilungen für die Strukturierung und Beschreibung von Maschineneigenschaften verwendet werden. Dadurch ist es möglich, die Sprachkonstrukte der Applikationsmuster von den Elementen im *Core*-Paket abzuleiten. Dies erfolgt durch die Definition entsprechender Artikulationsbeziehungen (siehe Definition 4-6).

Wenn im Folgenden von Modellelementen die Rede ist, so sind damit immer die Elemente einer Modellinstanz gemeint. Diese Präzisierung ist für das Verständnis wichtig, da in den Erläuterungen zum Metamodell häufig zwischen der Meta- und der Instanzebene gewechselt wird.

Es gibt zwei zentrale Elemente im Metamodell-Kern aus Abbildung 5-18. Mit dem Sprachkonstrukt *MTMComponent* wird der Komponenten-Begriff implementiert, der in den einzelnen fachspezifischen Modellbeschreibungen auf Instanzebene verwendet wird. Demnach

können ein Assembly oder ein Part durch das Sprachkonstrukt *MTMComponent* abgebildet werden. Weitere Ausprägungen der *MTMComponent* auf der Instanzebene sind beispielsweise die *Functionblocks* der IEC 61131-3 oder die Modell- und Klassendefinitionen aus Modelica (siehe Abschnitt 3.4.4). Eine *MTMComponent* erlaubt folglich die Abbildung der Hierarchisierungskonzepte, die in den einzelnen Modellierungssprachen vorhanden sind. Die Definition verschachtelter Komponentenstrukturen erfolgt anhand des passiven Sprachkonstrukts *Structure*. Dieses ermöglicht die Abbildung einer Beziehung zwischen *MTMComponent*-Elementen, wobei letztere die Rolle übergeordneter Komponenten *(group)* oder von Subkomponenten *(unit)* einnehmen können. Zur besonderen Kennzeichnung ist *Structure* mit dem Stereotype *assoc* ausgezeichnet, um zu verdeutlichen, dass dieses Element Beziehungen (Assoziationen) zwischen einzelnen Sprachkonstrukten beschreibt.

Das Element *MTMFunction* fasst alle Beschreibungsmittel zusammen, die für die Spezifikation der Funktionen einzelner Komponenten verwendet werden. Durch die Assoziation *perform* wird definiert, dass *MTMFunction*-Elemente *MTMComponent*-Elementen zugewiesen werden können. Das passive Sprachkonstrukt *Structure* ermöglicht auch für *MTMFunction* den Aufbau von Hierarchien im Sinne von Gesamt- und Teilfunktionen, wie dies in [BIRLI ET AL. 1997] für die Entwicklung mechatronischer Systeme gefordert wird.

Ein weiteres passives Sprachkonstrukt ist *ConnectableElement*. Auch damit können spezielle Beziehungen zwischen Sprachkonstrukten definiert werden. Es entspricht einer Generalisierung von *Energetic*, *Mechanical* und *Information*. Diese drei Sprachkonstrukte korrespondieren mit den *Stoff-*, *Energie-* und *Informationsfluss*-Definitionen aus [PAHL ET AL. 2004], [KOLLER 1985] und [ROTH 2001]. Dieser Sachverhalt wird mit dem Stereotype *flow* verdeutlicht.

Ein *ConnectableElement* beschreibt die Relation zweier *EObject*-Konstrukte. Ein *EObject* stellt ein allgemeines Element einer Modellbeschreibung. Es besitzt die Attribute *id* für die eindeutige Referenzierung und *name* für die Bezeichnung von Modellelementen. Wie aus Abbildung 5-18 zu entnehmen ist, ist *EObject* eine Generalisierung der Elemente *MTMFunction* und *MTMComponent*. Diese Elemente übernehmen daher neben den Attributen von *EObject* auch dessen Assoziations-Beziehungen zu *ConnectableElement.* Damit ist es beispielsweise möglich, komplexe Funktionsstrukturen nach [PAHL ET AL. 2004], [KOLLER 1985] und [ROTH 2001] abzubilden. Darüber hinaus sind damit die wesentlichen Grundlagen vorhanden, um die Methoden nach [KALLMEYER 1998] und [FLATH 2002] modelltechnisch zu erfassen. Hierfür ist jedoch die Definition weiterer Sprachkonstrukte in einem gesonderten Applikationsmuster notwendig.

Das Sprachkonstrukt *Port* erlaubt die Abbildung einer Schnittstelle. Schnittstellen in diesem Sinne können Software-Interfaces, aber auch Steckverbindungen von elektrischen Betriebsmitteln sein. Jeder *Port* gehört zu exakt einer *MTMComponent*, jedoch müssen diese nicht zwingend einen *Port* besitzen. Da ein *Port*-Element einem *MTMComponent*-Element zugeordnet ist, können über ein *ConnectableElement*-Sprachkonstrukt Beziehungen zwischen *Port*-Elementen aufgebaut werden. Ein *ConnectableElement* kann einen Energie-, Mechanik-, oder Informationsfluss darstellen. Entsprechend verarbeitet der *Port* als Schnittstelle unterschiedliche Arten von ein- und ausgehenden Fluss-Elementen, beispielsweise Strom (Energie), Werkstücktransport (Mechanik) oder Sensorsignale (Information). Das Sprachkonstrukt

FlowElement bildet diesen Sachverhalt ab, indem es als Platzhalter für aktive Sprachelemente dient, welche in den einzelnen Applikationsmustern definiert werden. Für die Beschreibung der spezifischen Fluss-Elemente sind diese den entsprechenden Sprachkonstukten *ConnectableElement* und *FlowElement* zuzuordnen. Dadurch wird verhindert, dass auf Instanzebene verschiedene Komponenten mit unterschiedlichen Schnittstellenbeschreibungen verknüpft werden können.

Durch das Element *Constraint* können den Sprachkonstrukten *MTMFunction* und *MTMComponent* Bedingungen zugeordnet werden. Darunter fallen alle Modellelemente, die einer Funktions- oder Komponentenbeschreibung zusätzliche charakterisierende Eigenschaften zuweisen. Ein *Constraint* könnte beispielsweise durch eine Anforderung im Lasten- bzw. Pflichtenheft formuliert werden und für die Funktion „Schutztür öffnen“ einen maximalen Zeitwert vorgeben. Darunter fallen auch Toleranzangaben für Bauteile oder die Berücksichtigung bestimmter Normen. Die Arten und Formen von *Constraint*-Elementen auf der Modellinstanzebene sind vielfältig und hängen von der jeweiligen Modellierungssprache ab, in der sie definiert sind. Durch das Attribut *kind* vom Typ *ConstraintKind* können prinzipiell drei verschiedene Arten von Constraints definiert werden. Soll mit dem Sprachkonstrukt *Constraint* eine Vorbedingung ausgedrückt werden, besitzt das Attribut *kind* den Wert *pre*. Eine Vorbedingung im Rahmen einer Verhaltensbeschreibung legt fest, dass eine bestimmte Ausgangsvoraussetzung zum Zeitpunkt eines Funktionsaufrufs oder einer Ausführungsinitialisierung erfüllt sein muss. Entsprechend wird mit dem Wert *post* eine Nachbedingung definiert. Mit *invariant* wird festgelegt, dass eine Bedingung unabhängig vom Zeitpunkt gültig sein muss. Nicht immer sind alle drei Arten von Constraints zulässig. Welche Arten verwendet werden, muss in den Applikationsmustern gesondert beschrieben werden.

Die Elemente *Applicationpattern* und *View* werden vorwiegend von den XML-Adaptern zur Einordnung der Sprachelemente in bestimmte Applikationsmuster und Sichten verwendet. Sie dienen dazu, den Modellkontext sowie den Anwendungsbereich zu beschreiben.

5.4.3 Die Applikationsmuster

Die Applikationsmuster definieren die Sprachkonstrukte für die in Abschnitt 5.3 vorgestellte Vorgehensweise und Modellierungstechnik. Für die Erstellung der Klassendiagramme wurden, gemäß den Erläuterungen in Abschnitt 5.2, YAT-Modelle zur Erörterung der zu integrierenden Daten angefertigt. Im Folgenden werden die einzelnen Applikationsmuster vorgestellt sowie deren Aufgabe im Rahmen der virtuellen Werkzeugmaschine erläutert. Sprachkonstrukte, die nicht im aktuellen Applikationsmuster definiert sind, werden entsprechend durch Grautöne hervorgehoben. Die Kennzeichnung der einzelnen Applikationsmuster beruht auf einer Abkürzung des vollen Namens.

Für die Beschreibung der einzelnen Applikationsmuster werden folgende charakterisierende Eigenschaften verwendet:

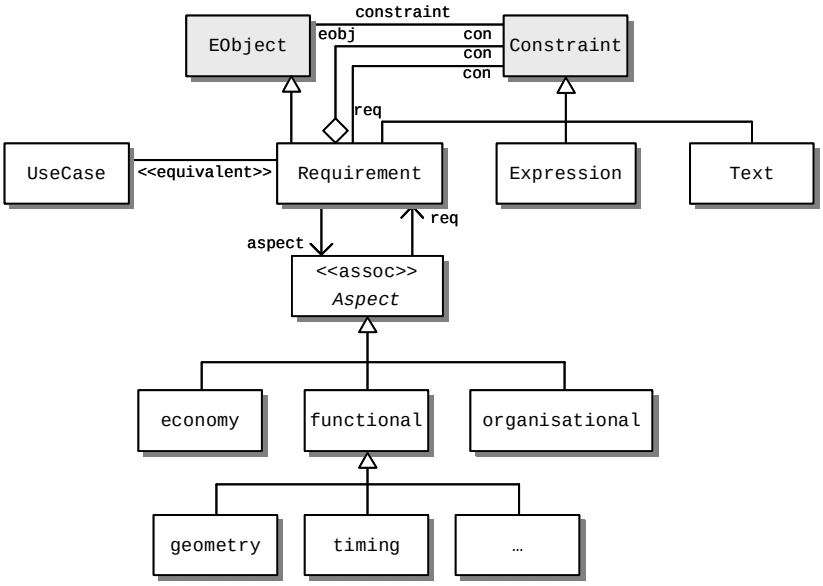
- ***Struktur:*** UML-Klassendiagramm zur Abbildung der Sprachkonstrukte des Applikationsmusters

- ***Name:*** Namenskürzel der vollen Bezeichnung
- ***Aufgabe:*** Zielsetzung und Aufgabe des Applikationsmusters im Rahmen der virtuellen Werkzeugmaschine
- ***Modellbezug:*** Referenz auf das Modell bzw. Softwarewerkzeug im Entwicklungsprozess
- ***Beschreibung:*** Erläuterung der Sprachkonstrukte und deren Abhängigkeiten
- ***Anwendung:*** Einsatzbereich und Zweck der Sprachkonstrukte aus der Sicht des Anwenders
- ***Abgeleitet von:*** Grundlegende Modellkonzepte, Modellierungstechniken, Normen und Datenformate, welche die Basis für die Formulierung des Applikationsmusters bilden.

Im Folgenden sind die einzelnen Applikationsmuster in tabellarischer Form und auf der Grundlage der vorgestellten Eigenschaften aufgeführt.

Applikationsmuster Requirementsengineering (REQ)

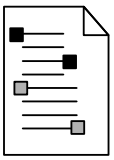
Struktur:



Name: REQ (REQuirementsengineering)

Aufgabe: Erfassen und Klassifizieren von Anforderungen aus einem Lasten- und Pflichtenheft

Modellbezug: Lasten- und Pflichtenheft, siehe Abbildung 5-9 (Seite 95)



Beschreibung:

- Das Element Requirement ist durch eine Artikulationsbeziehung mit EObject verbunden und kann somit anderen Sprachkonstrukten, die EObject zugeordnet sind, zugewiesen werden.
- Darüber hinaus ist Requirement auch dem Element Constraint zugeordnet und kann somit zur Charakterisierung oder zur Definition von Randbedingungen für Elemente von MTMFunction und MTMComponent verwendet werden.
- Die Sprachkonstrukte Expression und Text kennzeichnen zwei Möglichkeiten zur Formulierung von Constraint-Elementen. Durch die Aggregations-Beziehung können Requirement-Sprachkonstrukte grundsätzlich derartige Elemente enthalten.

- Das passive Sprachkonstrukt Aspect erlaubt die Formulierung eines speziellen Betrachtungsfokus auf das zu entwickelnde System. Aspekte ermöglichen somit eine Einordnung und Gliederung von Anforderungen. Dem Element Aspect sind entsprechende Kategorien zugeordnet.

- Das Sprachkonstrukt UseCase ermöglicht es, UML-UseCase-Diagramme für die Formulierung von Anforderungen einzusetzen. Das Element UseCase ist durch eine Äquivalenzbeziehung mit Requirement verbunden. Dadurch gelten die für Requirement gemachten Aussagen auch für das Element UseCase.

Anwendung:

- Klassifizierung, Gliederung und automatisierte Einordnung von Anforderungen in Office-Dokumente

- Verknüpfen der Anforderungen mit den Funktionen und Komponenten des zu entwickelnden Systems

Abgeleitet von:

- Gliederung des Pflichtenhefts nach [Aßmann 1996]

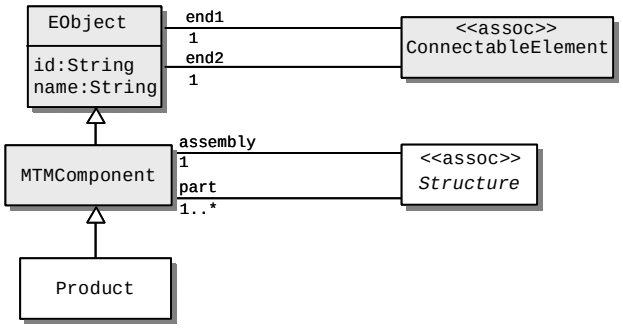
- Datenmodell Pflichtenheft nach [Uhlig 2002]

- Methode und Klassifizierung nach [Deifel 2001] und [Geisberger 2005]

- Open Document Format for Office Applications
  [ISO/IEC DIS 26300 2006]

Applikationsmuster Mechanikkonstruktion (MCAD)

Struktur:



Name: MCAD (Mechanical Computer Aided Design)

Aufgabe: Abbilden von Produktstrukturen im CAD-Bereich

Modellbezug: Produktstruktur aus dem 3D-CAD-Modell, siehe Abbildung 5-8 (Seite 94)

Beschreibung:

- Produktstrukturen werden in der Mechanikkonstruktion anhand der dort erstellten 3D-CAD-Modelle festgelegt. Die einzelnen Baugruppen und Bauteile werden in einer Datei nach ISO 10303-21 anhand des Modellelements Product definiert. Demzufolge wird dieses als Sprachkonstrukt in das vorliegende Applikationsmuster aufgenommen und mittels einer Artikulationsbeziehung dem Sprachkonstrukt MTMComponent zugeordnet.
- Durch das passive Sprachkonstrukt Structure, welches zwei MTMComponent-Sprachkonstrukte miteinander verknüpft, können komplexe Produktstrukturen eines STEP-Modells abgebildet werden.

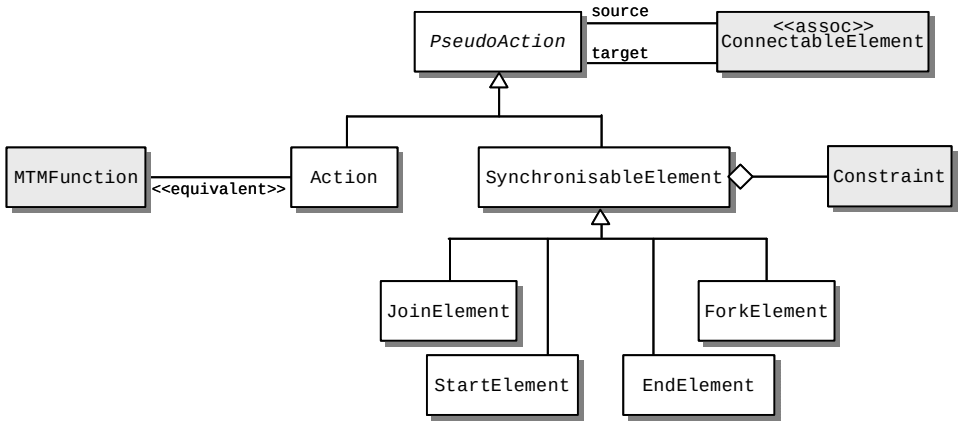
Anwendung: Aufbau der Produktstruktur aus einer STEP-Datei, bestehend aus *Assembly*- und *Product*-Definitionen

Abgeleitet von:

- Serie 21 der ISO 10303 [ISO 10303-21 2002]
- Applikationsprotokoll ISO 10303-203 [ISO 2004]

Applikationsmuster Aktivitätsdiagramm (FSTRUC)

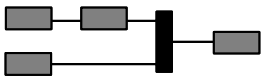
Struktur:



Name: FSTRUC (Function STRUCture)

Aufgabe: Beschreiben von Gesamt- und Teilfunktionen, Aufbau von Funktionsstrukturen

Modellbezug: Aktivitätsdiagramm, siehe Abbildung 5-10 (Seite 96)



Beschreibung:

- Das abgebildete Klassendiagramm stellt einen vereinfachten Ausschnitt aus dem UML-Metamodell für die Definition von Aktionen in Aktivitätsdiagrammen dar.
- Im Rahmen der in Abschnitt 5.3.2 vorgestellten Modellierungstechnik entspricht das Element Action einem Sprachkonstrukt MTMFunction.
- Action-Elemente werden ähnlich der blockorientierten Modellierung (siehe Abschnitt 3.4.4) miteinander verknüpft. Der Aufbau derartiger Strukturen wird durch das passive Sprachkonstrukt ConnectableElement aus dem Metamodell-Kern ermöglicht. Es verbindet PseudoAction-Sprachkonstrukte. Action-Elemente sind PseudoAction-Sprachkonstrukten zugeordnet.
- Entsprechend der blockorientierten Modellierung bestimmt die Reihenfolge die Berechnungsvorschrift bzw. die funktionale Abfolge. In diesem Sinne übernehmen Action-Elemente die Rolle einer Quelle (source) bzw. Senke (target).

- Es ist allerdings möglich, parallele Ablaufstrukturen von Action-Elementen zu erstellen. Für die Koordinierung sind entsprechende SynchronisableElement-Sprachkonstrukte vorgesehen.

Anwendung:

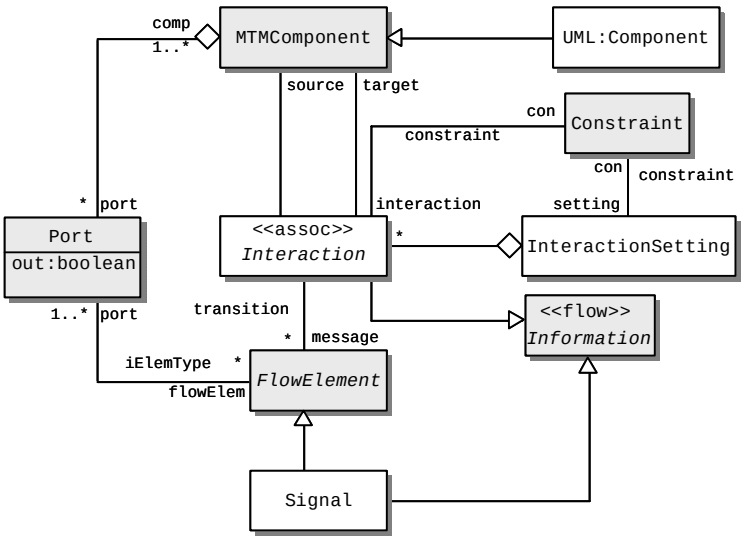
- Abbilden von Funktionsstrukturen nach Konstruktionsmethoden aus der Mechanikentwicklung in Anlehnung an die Aktivitätsdiagramme aus der UML
- Funktionale Dekomposition der Systemeigenschaften in Gesamt- und Teilfunktionen

Abgeleitet von:

- Konstruktionsmethoden nach [Pahl et al. 2004], [Koller 1985] und [Roth 2001]
- Funktionale Gliederung nach [Birli et al. 1997]
- Aktivitätsdiagramm, UML-Metamodell [OMG 2006b] und [Born et al. 2004]

Applikationsmuster Sequenzdiagramm (SEQ)

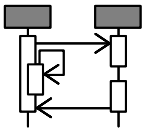
Struktur:



Name: SEQ (SEQuenzdiagramm)

Aufgabe: Abbilden der Interaktionen der Baugruppen und Komponenten sowie Definition von Schnittstellen aus der Sicht der Steuerungstechnik

Modellbezug: Sequenzdiagramm, siehe Abbildung 5-11 (Seite 97)



Beschreibung:

- Zur Beschreibung einer Interaktion zwischen verschiedenen Komponenten wird das passive Sprachkonstrukt Interaction verwendet. Es beschreibt den Informationsaustausch zwischen zwei MTMComponent-Elementen für spezifische Steuerungsfunktionen. Aus diesem Grund ist Interaction dem Sprachelement Information zugeordnet.

- Das Sprachkonstrukt Signal ist FlowElement und Information zugeordnet und stellt somit ein zulässiges Fluss-Element für das Interaktionsdiagramm dar.

- Nach der in Abschnitt 5.3.2 erläuterten Funktionsmodellierung werden Interaktionsdiagramme mit UML-Case-Tools erzeugt. Aus diesem Grund ist das Sprachkonstrukt UML:Component der MTMComponent zugeordnet.

- Einer Interaktion zwischen zwei MTMComponent-Sprachkonstruk-

ten können Constraints zugewiesen werden. Dadurch ist es möglich, beispielsweise Werte für die Zeitüberschreitung (Timeout) von Interaktionen anzugeben. Das Sprachkonstrukt InteractionSetting erlaubt es, mehrere Interaktionen zusammenzufassen und dieser Gruppe bestimmte Contraints zuzuordnen. Zum Beispiel können so die einzelnen Interaktionen für die Steuerungsfunktion „Werkzeug wechseln“ zusammengefasst und die gültige Dauer des Vorgangs als Constraint definiert werden. Es sind alle drei Arten von Constraints möglich.

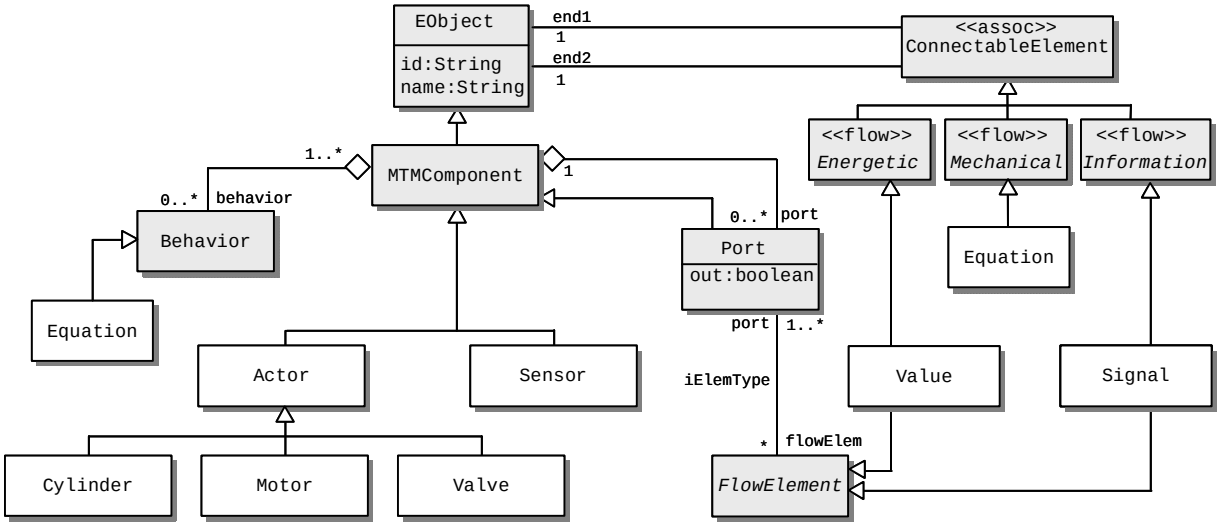
Anwendung:

- Spezifizieren der Steuerungsfunktionen durch die Abbildung der Signalfolgen zwischen den Steuerungskomponenten
- Verlinken funktionaler Anforderungen an die Steuerungsentwicklung durch Constraints

Abgeleitet von: Interaktionsdiagramm, UML-Metamodell [OMG 2006b] und [Born et al. 2004]

Applikationsmuster Technologieschema (TEC)

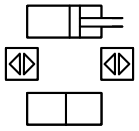
Struktur:



Name: TEC (TEChnology Diagramm)

Aufgabe: Darstellen einer Prinzipskizze mit den wichtigsten Komponenten

Modulbezug: Technologieschema, siehe Abbildung 5-12 (Seite 98)



Beschreibung:

- Dieses Applikationsmuster beschreibt die zentralen Komponenten einer Werkzeugmaschine, deren Schnittstellen sowie ihr Systemverhalten.
- Die Sprachkonstrukte Actor und Sensor sind dem Sprachkonstrukt MTMComponent zugeordnet. Das Sprachkonstrukt Actor ist ein aktives Sprachelement und ermöglicht die Abbildung allgemeiner Aktoren in einem System. Eine Konkretisierung von Actor erfolgt durch die Zuordnung der Sprachelemente Cylinder, Motor und Valve.
- Die Schnittstellenbeschreibung durch das Port-Sprachkonstrukt hängt vom ConnectableElement ab, mit dem verschiedene Komponenten verbunden sind. Die Verbindung zwischen einem Ventil, das einen Zylinder ansteuert, ist ein ConnectableElement vom Typ Energetic. Die Schnittstellenbeschreibung besteht in diesem Fall aus Sprachkonstrukten vom Typ FlowElement, die ConnectableElement zugeordnet sind. Das Metamodell stellt hierfür das Sprachelement Value zur Verfügung.

Der für die Steuerung des Zylinders erforderliche Druck kann somit durch eine einfache Konstante abgebildet werden. Die Schnittstellenbeschreibung des Sprachkonstruktes Port hängt also davon ab, durch welches ConnectableElement zwei Komponenten miteinander verbunden sind.

- Das Verhalten der MTMComponent-Sprachkonstrukte des Technologieschemas wird durch das Equation-Sprachelement definiert. Damit ist eine mathematische Beschreibung des Verhaltens gemeint.

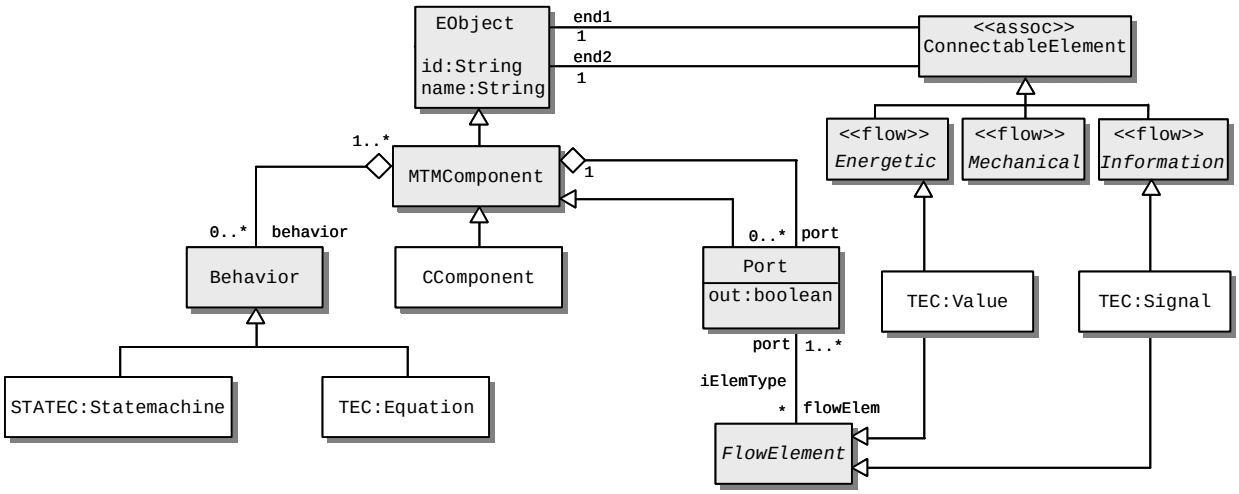
Anwendung:

- Abbilden der prinzipiellen Funktionsweise des Systems bzw. Subsystems
- Identifizieren der zentralen Automatisierungskomponenten und deren Interaktion durch eine Schnittstellenbeschreibung

Abgeleitet von: DIN ISO 1219 [DIN ISO 1219 2004]

Applikationsmuster Komponentendiagramm (COMPC)

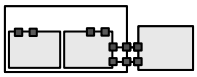
Struktur:



Name: COMPC (COMPonent Chart)

Aufgabe: Spezifikation der Struktur der Steuerungssoftware

Modulbezug: Komponentendiagramm, siehe Abbildung 5-13 (Seite 99) und Abbildung 5-15 (Seite 100)



Beschreibung:

- Im Komponentendiagramm werden vorwiegend Sprachkonstrukte aus dem Core-Paket verwendet. Mit dem Element CComponent wird eine steuerungstechnische Komponente definiert, die eine Steuerungsfunktion kapselt und über Port-Sprachelemente mit der Umgebung kommuniziert.

- CComponent ist dem Sprachkonstrukt MTMComponent zugeordnet. Für die Verhaltensbeschreibung stehen zwei Sprachelemente zur Verfügung. Mit TEC:Equation können mathematische Funktionen für die Beschreibung des Verhaltens einer CComponent verwendet werden. TEC:Equation ist im Applikationsmuster TEC definiert. Um es im Applikationsmuster COMPC verwenden zu können, muss das Sprachkonstrukt referenziert werden. Hierzu wird der Name des Applikationsmusters dem Namen des zu referenzierenden Sprachelements vorangestellt und durch einen Doppelpunkt getrennt.

- Mit dem Sprachkonstrukt STATEC:Statemachine ist eine diskrete, zu-

standsbasierte Verhaltensbeschreibung eines CComponent-Sprachelements möglich. Dieses Sprachkonstrukt wird im folgenden Applikationsmuster beschrieben.

- Durch die Referenzierung werden die Sprachkonstrukte nicht in das Applikationsmuster importiert. Deshalb ist es notwendig, die Sprachelemente TEC:Signal und TEC:Value ebenfalls zu referenzieren und damit festzulegen, dass eine CComponent über ein Port-Sprachelement durch Signa-l bzw. Value-Sprachkonstrukte kommuniziert.

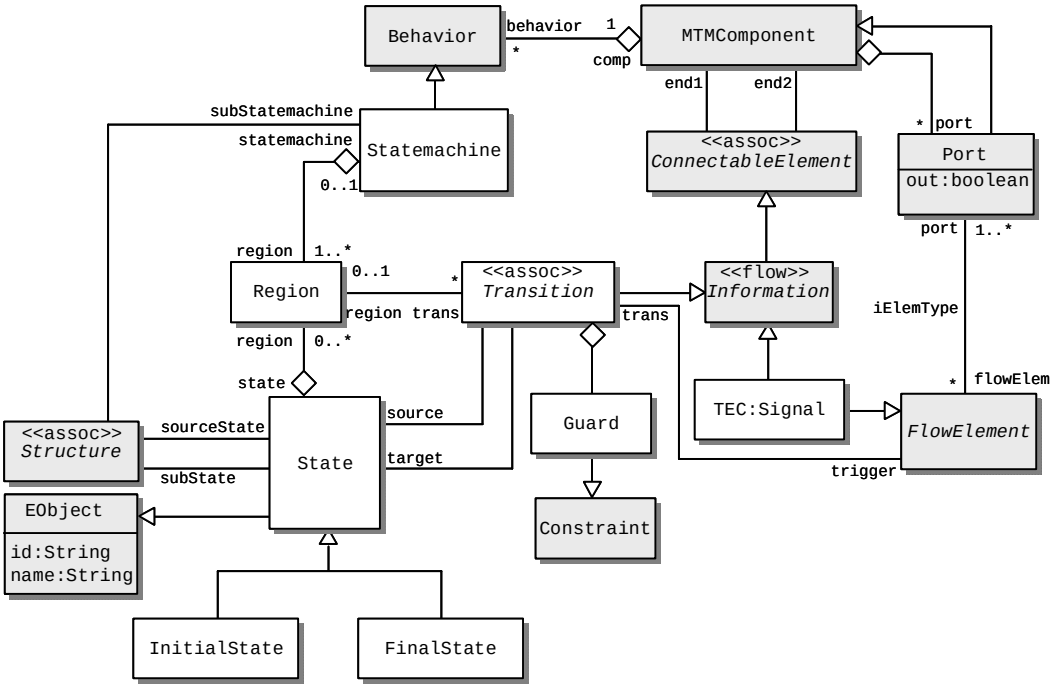
Anwendung:

- Ableiten von steuerungsrelevanten Komponenten aus bestehenden CAD-Modellen
- Aufbau eines Architekturmodells der Steuerungssoftware durch die UML

Abgeleitet von: Zustandsgraph, UML-Metamodell [OMG 2006b], [Born et al. 2004] und [Medvidovic et al. 2002] sowie XMI [OMG 2005]

Applikationsmuster Zustandsgraph (STATEC)

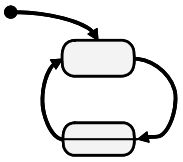
Struktur:



Name: STATEC (STATEmachine Chart)

Aufgabe: Abbilden des diskreten Verhaltens von Hard- und Softwarekomponenten

Modulbezug: Zustandsgraph, siehe Abbildung 5-14 (Seite 100)



Beschreibung:

- Das abgebildete Klassendiagramm stellt einen vereinfachten, an die Systemmodellierung aus Abschnitt 5.3 angepassten Ausschnitt des UML-Metamodells für Zustandsautomaten dar.
- Das Sprachkonstrukt Statemachine ist Behavior zugeordnet. Dadurch können MTMComponent-Sprachkonstrukte Zustandsgraphen zur Verhaltensbeschreibung besitzen.
- Region teilt analog zur UML einen Zustandsgraph oder einen Zustand in einzelne Bereiche auf. Dadurch ist es möglich, zueinander parallele Abläufe zu beschreiben.

- Die verschiedenen Zustände des Systems werden mit State beschrieben. Die Zustandsübergänge werden durch das passive Sprachkonstrukt Transition abgebildet. Eine Transition wird durchlaufen, wenn ein entsprechendes Tec:Signal den Zustandsübergang auslöst. Für ein deterministisches (vorhersagbares) Verhalten müssen zwei Transitionen, die denselben Ausgangszustand haben, durch verschiedene Signale ausgelöst werden.
- InitalState und FinalState bilden den Anfangs- und Endzustand ab.
- Mit dem Sprachelement Guard kann festgelegt werden, wann eine Transition durchlaufen werden soll und wann nicht. Ein Guard kann durch ein Constraint-Sprachkonstrukt definiert werden. Sind ein TEC:Signal und ein Guard an einer Transition gegeben, so wird die Transition nur dann ausgelöst, wenn die Auswertung des Guard-Sprachelements „True“ ergibt. Es sind alle drei Arten von Constraints möglich.
- Zustände können durch das passive Sprachkonstrukt Structure aus dem Core-Package hierarchisiert werden. Die Definition der source- und subStates ist hierbei analog zur UML.
- Mit Structure können auch Unterzustandsgraphen (subStatemachine im Klassendiagramm) beschrieben werden. Die Definition der Unterzustandsgraphen erfolgt analog zur UML.

Anwendung:

- Beschreiben der Systemzustände und der Zustandsübergangsbedingungen
- Beschreiben der Steuerungsfunktionen
- Abbilden des dynamischen Verhaltens von Hardware-Komponenten zur Systemspezifikation

Abgeleitet von: Zustandsgraph des UML-Metamodells [OMG 2006], [Born et al. 2004] und [Medvidovic et al. 2002]

Die erläuterten Applikationsmuster beziehen sich auf das in Abschnitt 5.3 beschriebene Konzept der Systemmodellierung für Werkzeugmaschinen. Die Strukturierung und Kennzeichnung sowie die Definition der Applikationsmuster selbst ändern sich entsprechend den im Entwicklungsprozess eingesetzten Modellierungstechniken und Softwarewerkzeugen. Die Definition eines firmenspezifischen Metamodells zur Datenintegration erfolgt nach der in Abschnitt 5.2.2 erläuterten Vorgehensweise zur Bestimmung der Applikationsmuster. Der Metamodell-Kern bleibt unverändert und wird anhand der in Definition 4-7 festgelegten Abbildungsfunktionen als Grundlage für die Definition anwendungsbezogener Sprachkonstrukte verwendet.

Anhand der im Metamodell hinterlegten fachbereichsübergreifenden Abhängigkeiten zwischen den einzelnen Modellierungselementen können die in den einzelnen Disziplinen erstellten Modelle zusammengeführt werden. Damit dieser Schritt möglich ist, müssen die proprietären Datenformate in ein auf das Metamodell abgestimmtes Zwischenformat konvertiert werden. Dieses Zwischenformat wird als *Virtual Machine Tool Language* (VMTL) bezeichnet. Im folgenden Abschnitt wird die VMTL erläutert und deren Bedeutung innerhalb der virtuellen Werkzeugmaschine in Bezug auf die Datenintegration hervorgehoben.

5.5 Die Virtual Machine Tool Language

Die fachbereichsspezifischen Softwarewerkzeuge speichern die Daten in einem proprietären Datenformat ab. Die XML-Adapter der virtuellen Werkzeugmaschine konvertieren diese Daten in die *Virtual Machine Tool Language* (VMTL). Die VMTL ist eine XML-Sprache, die auf den Sprachkonstrukt-Definitionen des Metamodells aufbaut. Alle Entwicklungsdaten, die vom Metadaten-Management-System verwaltet werden, liegen in der virtuellen Werkzeugmaschine als VMTL-Dateien vor und bilden somit die Instanzebene des Machine Tool Metamodells. In Abbildung 5-19 ist der prinzipielle Aufbau einer VMTL-Datei in Form einer Baumstruktur dargestellt.

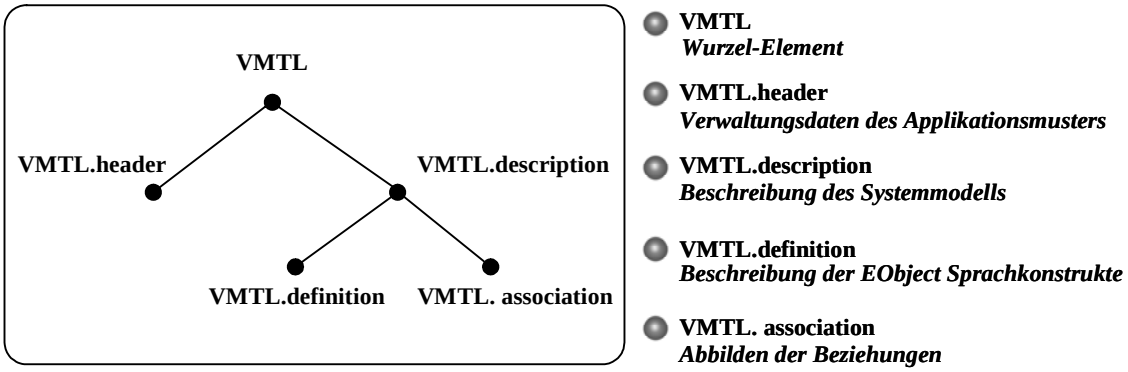


Abbildung 5-19: Aufbau einer VMTL-Datei

Das Wurzel-Element einer VMTL-Datei wird mit dem Element *VMTL* beschrieben. Hier werden durch das XML-Attribut *xmlns* (siehe Abschnitt 11.3.1, [W3C 2004A]) die Namensräume definiert, in denen die weiteren XML-Elemente eingeordnet sind. Die Bezeichnungen der Namensräume entsprechen den Kürzeln der Applikationsmuster aus Abschnitt 5.4.3 sowie dem Metamodell-Kern aus Abschnitt 5.4.2. Der mit dem XML-Element *VMTL.header* umrahmte Bereich enthält Verwaltungsinformationen für das Metadaten-Management-System zur Organisation der einzelnen VMTL-Dateien. Unter anderem werden hier Informationen über das Softwarewerkzeug und die Version des XML-Adapters abgelegt.

Die eigentlichen Produktentwicklungsdaten befinden sich in dem Bereich, der mit dem XML-Element *VMTL.description* bezeichnet ist. Dieser Block ist in zwei Zweige gegliedert, die mit *VMTL.definition* und *VMTL.association* ausgezeichnet sind. Der mit *VMTL.definition* markierte Block enthält die Beschreibungen der *EObject*-Sprachkonstrukte aus den Abschnitten 5.4.2 und 5.4.3.

In diesem Bereich werden die *MTMComponent*, die *MTMFunction* sowie die diese beschreibenden Elemente aufgeführt. Die Beziehungen zwischen diesen Elementen werden in dem mit *VMTL.association* markierten Block beschrieben.

Im Folgenden wird anhand des Beispiels aus Abbildung 5-13 (Seite 99) und Abbildung 5-14 (Seite 100) der Aufbau der VMTL sowie der Zusammenhang mit dem Metamodell aus Abschnitt 5.4 erläutert. Das Beispiel zeigt das Softwaremodell des Werkzeugwechslers einer Werkzeugmaschine. In Abbildung 5-13 sind die Softwarearchitektur mit den zentralen Softwarekomponenten sowie die Schnittstellen zum physikalischen System zu sehen. Die Softwarekomponente *Werkzeugwechsler* enthält die Subkomponenten *Werkzeugspanner* und *Schutztür*. Die Funktionsweise der *Schutztür* ist in Abbildung 5-14 in Form eines Zustandsgraphen dargestellt. Dieses Softwaremodell wird im Informationsverbund der virtuellen Werkzeugmaschine mit dem UML Case-Tool *Poseidon* der Firma Gentleware [GENTLEWARE 2005] erstellt. Poseidon exportiert das Modell in das XMI-Format und dieses wird vom XML-Adapter der virtuellen Werkzeugmaschine in VMTL konvertiert und anschließend importiert. Im Folgenden wird die vom XML-Adapter erzeugte VMTL-Datei präsentiert.

In Abbildung 5-20 a ist das Wurzel-Element mit den in der Datei verwendeten Applikationsmustern als XML-Namensraumdefinitionen zu sehen. Jedes XML-Element ist eindeutig einem Namensraum zugeordnet, der sich aus den Applikationsmustern sowie dem Metamodell-Kern ableitet. In Abbildung 5-20 b sind die Verwaltungsinformationen dargestellt. Unter anderem wird hier das Softwarewerkzeug Poseidon sowie das *COMPC* als Kennzeichnung für das Applikationsmuster aufgeführt. Die Angaben der Versionsnummern für das Softwarewerkzeug und das Importformat ermöglichen dem Metadaten-Management-System die Auswahl geeigneter XML-Adapter.

```
<VMTL xmlns:COMPC="http://www.iwb.tum.de/COMPC"
      xmlns:CORE="http://www.iwb.tum.de/CORE"
      xmlns:STATEC="http://www.iwb.tum.de/STATEC"
      xmlns:TEC="http://www.iwb.tum.de/TEC"
      vmtl.version="0.1">
```

a) Wurzelelement

```
<VMTL.header>
    <VMTL.mtm_prefix>COMPC</VMTL.mtm_prefix>
    <VMTL.toolName>Poseidon</VMTL.toolName>
    <VMTL.importFormat>XMI 2.0</VMTL.importFormat>
    <VMTL.toolVersion>3.01</VMTL.toolVersion>
</VMTL.header>
```

b) Verwaltungsinformationen

Abbildung 5-20: Wurzel-Element (a) und Verwaltungsinformation (b) einer VMTL-Datei

Der VMTL-Code in Abbildung 5-21 zeigt einen Auszug aus dem Definitionsteil. Hier sind die Softwarekomponenten *Werkzeugwechsler*, *Werkzeugspanner* und *Schutztür* enthalten. Für die Beschreibung wird das Sprachkonstrukt *CComponent* aus dem Applikationsmuster *COMPC* verwendet. Die Abbildung der Elemente aus dem Importformat XMI in die geeigneten Zielelemente der VMTL erfolgt durch den XML-Adapter, welcher auf die im Metamodell enthaltenen Abbildungsregeln zurückgreift. XML-Elemente, auf die innerhalb der VMTL-Datei referenziert werden soll, enthalten ein XML-Attribut *id*. Die *id* ist innerhalb der VMTL-Datei eindeutig und wird vom XML-Adapter automatisch erzeugt. Die systemweite Eindeutigkeit der einzelnen Elemente wird durch eine *Betriebsmittelkennzahl* (BMK) gewährleistet, die im XML-Attribut *name* als Präfix dem eigentlichen Namen des jeweiligen Elements vorangestellt wird.

```
…
<VMTL.description>
    <VMTL.definition>
     <COMPC:CComponent id="#201603" name="M21_MTSC:Werkzeugwechsler">
      <!--component properties-->
     </COMPC:CComponent>
     <COMPC:CComponent id="#201604" name="M81_MTSC:Werkzeugspanner">
      <!--component properties-->
     </COMPC:CComponent>
     <COMPC:CComponent id="#201605" name="M51_MTSC:Schutztuer">
      <!--component properties-->
      <STATEC:Statemachine id="#401000">
       <STATEC:Region id="#401001">
       </STATEC:Region>
      </STATEC:Statemachine>
     </COMPC:CComponent>
     …
     <STATEC:InitialState id="#402000"/>
     <STATEC:State id="#402001" name="ST_34_Tuer zu"/>
     …
     <STATEC:State id="#402004" name="ST_37_Tuer schliessen"/>
     <TEC:Signal id="#70500" name="ST_TC_12_oeffnen"/>
     …
     <TEC:Signal id="#70503" name="ST_TC_15_Time trigger"/>
   </VMTL.definition>
 <VMTL.description>
 …
```

Abbildung 5-21: Softwarekomponenten und Zustandsgraph in VMTL-Notation

Das Verhalten der Softwarekomponente *Schutztür* wird mit einem Zustandsgraph beschrieben. Im Metamodell wird das Verhalten über eine Aggregationsbeziehung den *MTMComponent*-Sprachkonstrukten zugewiesen. In VMTL wird dies dadurch abgebildet, dass die entsprechende Beschreibung des Zustandsgraphen als Kindelement dem jeweiligen XML-Element, welches das *MTMComponent*-Sprachkonstrukt auf Instanzebene widerspiegelt, hinzugefügt wird. Entsprechendes gilt für die in Abbildung 5-21 aufgeführten XML-Elemente *STATEC:InitialState*, *STATEC:State* und *TEC:Signal*.

Die Beziehungen zwischen den einzelnen Elementen sind im *VMTL.association*-Block definiert. Sie können prinzipiell durch passive Sprachelemente oder durch Aggregationsbeziehungen abgebildet werden. In Abbildung 5-22 a ist die Struktur der Softwarekomponenten zu sehen. Strukturen werden durch das im Metamodell-Kern (*Core*-Package) definierte passive Sprachelement *Structure* festgelegt. Für die Beschreibung einer Aggregation („Besteht aus“-Beziehung) zwischen verschiedenen Komponenten werden die im Metamodell definierten Rollen verwendet, die *MTMComponent* Sprachkonstrukte einnehmen können. *Assembly* bezeichnet dabei die Komponente, die aus mehreren Teilen besteht, mit *Part* werden die einzelnen Teile identifiziert. Mit dem XML-Attribut *idref* wird auf das jeweilige XML-Element verwiesen, das die entsprechende Komponente beschreibt.

```
…
<VMTL.association>
    <CORE:Structure>
        <CORE:Structure.assembly idref="#201603"/>
        <CORE:Structure.part idref="#201604"/>
        <CORE:Structure.part idref="#201605"/>
    </CORE:Structure>
…
```

a) Aufbau von Strukturen

```
…
  <STATEC:Transition id="104" idref="#401001">
     <STATEC:Transition.source idref="#402004"/>
     <STATEC:Transition.target idref="#402001"/>
     <STATEC:Transition.trigger idref="#70503"/>
  </STATEC:Transition>
 </VMTL.association>
</VMTL.description>
</VMTL>
```

b) Beschreibung von Beziehungen

Abbildung 5-22: Abbilden von Strukturbeziehungen und Relationen

Die Abbildung der Zustandsübergänge erfolgt analog. Wie in Abbildung 5-22 b zu sehen, wird durch das in STATEC definierte passive Sprachelement *Transition* eine Beziehung zwischen zwei Zuständen abgebildet. Durch die Rollennamen *Source* und *Target* wird die Richtung der Transition festgelegt. Relationen, die im Metamodell durch eine einfache Assoziation und ohne ein gesondertes passives Sprachelement beschrieben sind, werden durch die Rollennamen und das referenzierende Sprachkonstrukt wiedergegeben. Ein Beispiel hierfür sind die Triggersignale der Transitionen. In Abbildung 5-22 b werden die Signale, die einen Zustandsübergang auslösen, mit dem XML-Element *Transition.trigger* und dem XML-Attribut *idref* referenziert.

Um das Beispiel nicht durch redundante Details zu verkomplizieren, sind nicht alle Einzelheiten aus Abbildung 5-13 in der VMTL-Datei enthalten. So wurde auf eine Abbildung der *Ports* verzichtet. Deren Darstellung und Zuweisung zu den Komponenten erfolgt aufgrund der Aggregationsbeziehung (siehe Abbildung 5-18) analog zum Zustandsgraphen aus Abbildung 5-21. Ebenso wurden nur die für die Erläuterung wesentlichen Attribute berücksichtigt.

5.6 Funktionsweise der Integration von Entwicklungsdokumenten

Im Informationsverbund der virtuellen Werkzeugmaschine werden die Entwicklungsdaten in Form von Dokumenten bzw. Modellen mithilfe fachbereichsspezifischer Softwarewerkzeuge generiert. Anschließend werden diese Daten in die virtuelle Werkzeugmaschine importiert und durch die XML-Adapter in die neutrale VMTL konvertiert.

Wie in Abschnitt 11.3.1 erläutert, besitzen XML- bzw VMTL-Dateien eine baumförmige Struktur, welche durch die Anfragesprache *XPath* adressiert werden kann. XPath stellt damit die Implementierung der in Definition 4-4 vorgestellten Datenanfrage-Sprache dar. Die Integration der Daten aus den verschiedenen Fachbereichen erfolgt in der virtuellen Werkzeugmaschine durch das Zusammenfügen und Umhängen einzelner Äste oder ganzer Sub-Bäume. Geeignete XPath-Ausdrücke adressieren dabei Fragmente einzelner VMTL-Dateien, die sich anschließend zu neuen VMTL-Dateien zusammenfügen lassen. Das Beispiel in Abbildung 5-23 zeigt die Integration von Requirement-Beschreibungen in die VMTL-Datei aus Abbildung 5-21.

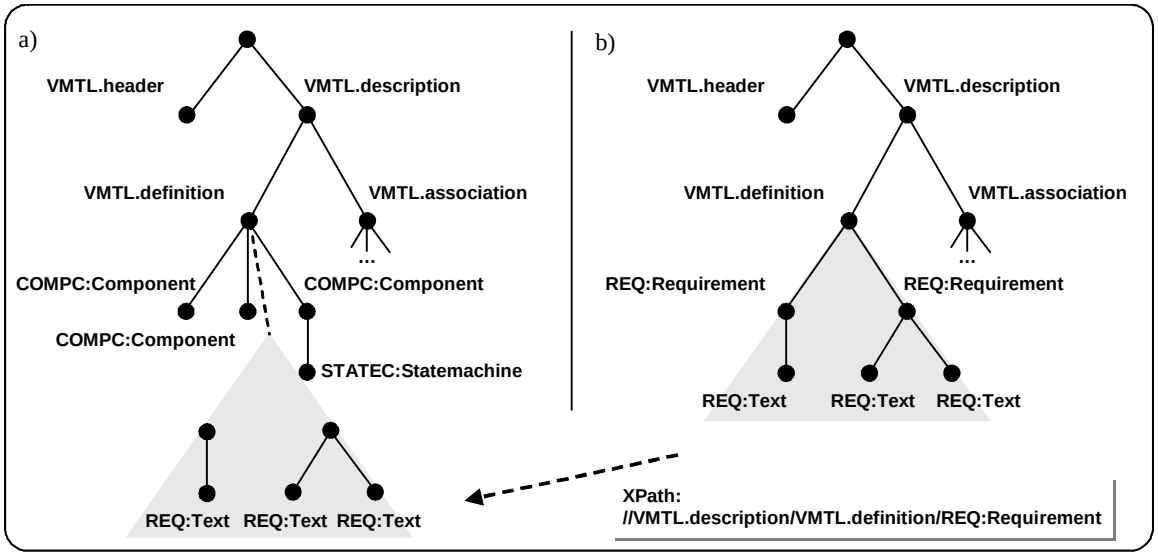


Abbildung 5-23: Zusammenfügen einzelner Äste oder Sub-Bäume zu neuen VMTL-Dateien

In Abbildung 5-23 b ist ein VMTL-Baum mit einzelnen Requirement-Textbausteinen zu sehen, die bei der Systemanalyse aus dem Lastenheft erstellt wurden. Um eine Nachverfolgbarkeit der Anforderungen zu ermöglichen, muss gewährleistet sein, dass die Textbausteine mit den entsprechenden Hard- und Softwarekomponenten verknüpft werden können. Dies erfolgt dadurch, dass mit einem XPath-Ausdruck alle Requirement-Beschreibungen adressiert und anschließend in den *VMTL.definition*-Block aus Abbildung 5-23 a eingefügt werden. Bei der Formulierung komplexerer XPath-Ausdrücke können auch die Attribute der Sprachkonstrukte, die im Metamodell definiert sind, verwendet werden. Nähere Informationen über XPath-Ausdrücke sind in [CLARK & DEROSE 1999] zu finden. Das Beispiel aus Abbildung 5-23 a zeigt den Ausschnitt einer VMTL-Datei, in der eine Struktur aus Softwarekomponenten und Verhaltensbeschreibungen in Form von Zustandsgraphen beschrieben ist. Die erweiterte VMTL-Datei wird durch geeigente XML-Adapter in das Zielformat exportiert und anschließend durch ein geeignetes Softwarewerkzeug der virtuellen Werkzeugmaschine eingelesen. Dieses erlaubt es, die Verknüpfung der Requirements mit den Softwarekomponenten zu erstellen und diese Information ebenfalls wieder als VMTL-Datei zu exportieren. Das Metadaten-Management-System besitzt eine graphische Benutzeroberfläche, um den gezielten Export von Entwicklungsdaten und deren Integration in ausgewählte Zielformate zu steuern.

Das beschriebene Gesamtszenario ist in Abbildung 5-24 veranschaulicht. Die Darstellung der Terminologie erfolgt durch eine Baumdarstellung, wobei die Wurzel die Bezeichnung des Zielformats symbolisiert. Die erste Blatt-Ebene eines jeden Asts repräsentiert dabei das Sprach- bzw. Modellkonstrukt dar, in das transformiert werden soll. Alle weiteren Begriffe sind Elemente der im Metamodell definierten Sprachkonstrukte für die virtuelle Werkzeugmaschine.

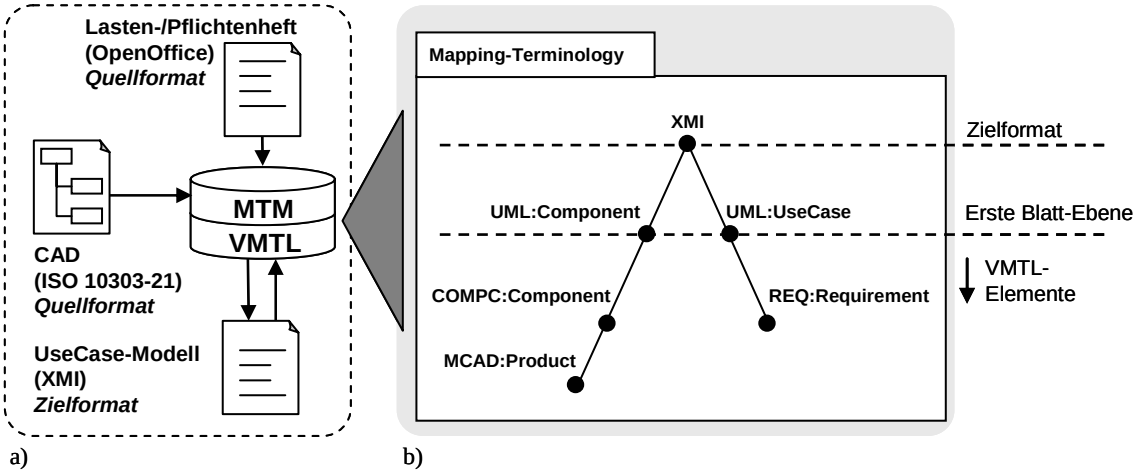


Abbildung 5-24: Definieren der Zielformate für die Transformation der Sprachelemente des Machine-Tool-Metamodells

Die Quelldaten liegen in verschiedenen proprietären Datenformaten vor. Im Beispiel aus Abbildung 5-24 a werden die zentralen Systemkomponenten aus dem 3D-CAD-Modell ausgelesen und in VMTL konvertiert. Die Anforderungen sind im Lasten- bzw. Pflichtenheft aufgeführt und werden als einzelne Textbausteine ebenfalls in VMTL umgewandelt. Die Transformation vom Quellformat in VMTL übernehmen die XML-Adapter der virtuellen Werkzeugmaschine auf der Grundlage der im Metamodell definierten Sprachkonstrukte. Für den Export der VMTL-Dateien müssen Regeln vorhanden sein, die bestimmen, welche Sprachkonstrukte in ein bestimmtes Zielformat transformiert werden können. Hierzu wird im Metamodell ein zusätzliches *Mapping-Terminology*-Paket definiert (Abbildung 5-24 b). Hier wird festgelegt, welche Sprachkonstrukte des Metamodells in welcher Form ins Zielformat konvertiert werden können. In Abbildung 5-24 b sind beispielsweise die Sprach-konstrukte *MCAD:Product* bzw. *COMPC:Component* als *UML:Component* in das Ziel-format XMI transformierbar. Analog dazu erfolgt die Konvertierung des Sprachkonstrukts *REQ:Requirement* als *UML:UseCase* in das XMI-Format. Auf dieser Grundlage kann der XML-Adapter anschließend den Export der Daten durchführen. Der in Abbildung 5-17 skizzierte Aufbau des Metamodells wird deshalb um ein zusätzliches Paket erweitert, so dass sich die in Abbildung 5-25 dargestellte Struktur ergibt.

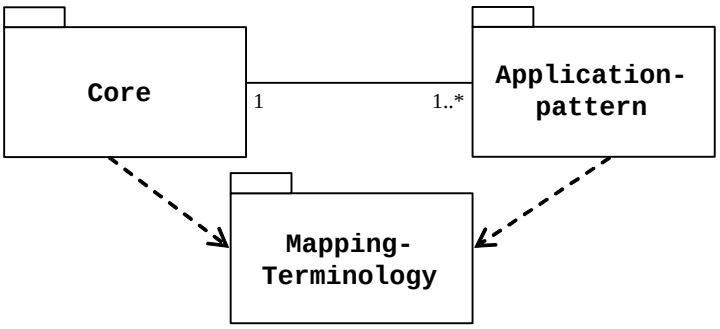


Abbildung 5-25: Aufbau des Machine-Tool-Metamodells mit Mapping-Paket

Im folgenden Kapitel wird nun die konkrete Umsetzung der diskutierten Konzepte vorgestellt. Hierzu werden die Architektur und die Implementierung des Metadaten-Management-Systems in Form eines Softwareprototyps beschrieben und anhand entsprechender Architekturmodelle veranschaulicht.

6 Architektur des Metadaten-Management-Systems

6.1 Übersicht

Im vorliegenden Kapitel wird die Implementierung der vorgestellten Konzepte als Softwareprototyp erläutert. Das Metadaten-Management-System bildet die Kernkomponente der virtuellen Werkzeugmaschine und setzt sich aus insgesamt drei Schichten zusammen (siehe Abbildung 6-1). Der Mediator aus Definition 4-7 ist in der gleichnamigen Mediator-Schicht umgesetzt. Wie in Abbildung 4-2 ersichtlich, ist das Metadaten-Management-System als Modul eines Applikationsservers ausgeführt.

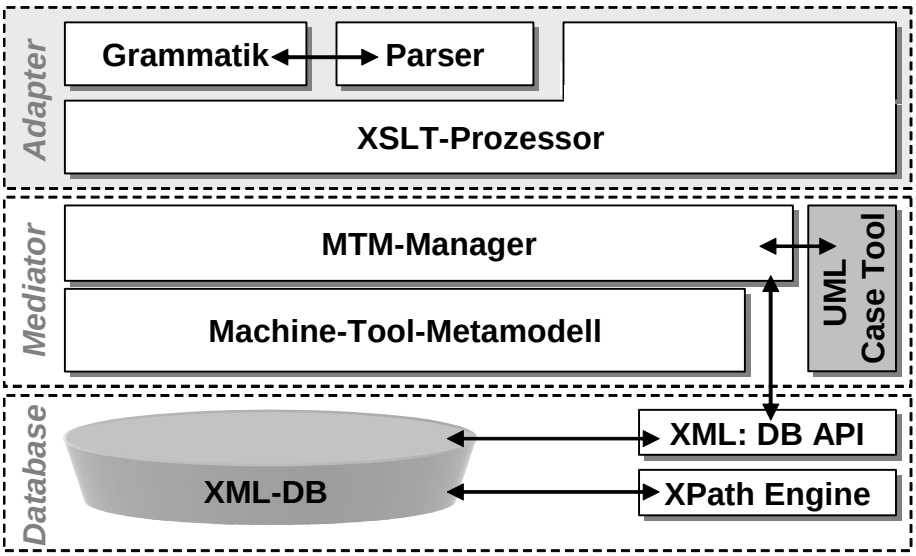


Abbildung 6-1: Architektur des Metadaten-Management-Systems

Die für den Aufbau einer Client-Server-Umgebung erforderliche Kommunikationsinfrastruktur wird vom Framework des Applikationsservers zur Verfügung gestellt und kann somit vom Metadaten-Management-System genutzt werden. Im Rahmen der Arbeit wird dieser Aspekt deshalb nicht weiter ausgeführt. In den folgenden Abschnitten werden die einzelnen Komponenten des Metadaten-Management-Systems beschrieben.

6.2 Die Adapter-Schicht

Diese Schicht stellt geeignete Schnittstellen zur Verfügung, um die verschiedenen Softwarewerkzeuge in den Informationsverbund der virtuellen Werkzeugmaschine einzubinden. Für jedes Softwarewerkzeug bzw. für jede Modellierungstechnik ist ein entsprechender Adapter erforderlich, der die Daten einliest und für die Weiterverarbeitung innerhalb der virtuellen Werkzeugmaschine aufbereitet. Die Adapter setzen sich aus einem Parser und einem Konvertermodul zusammen. Der Parser liest ein Dokument ein und gibt es an das Konvertermodul weiter. Dieses transformiert es zur internen Weiterverarbeitung in die VMTL. Der generelle Aufbau und der prinzipielle Funktionsablauf sind in Abbildung 6-2 zu sehen. Als Eingabeparameter sind dem Adapter das Dokument selbst und die Angabe des Dokument-Typs zu übergeben.

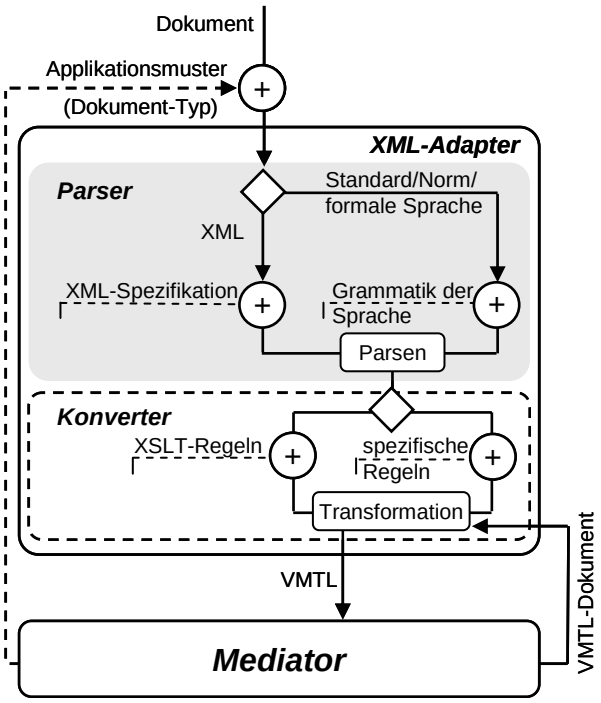


Abbildung 6-2: Funktionsweise der XML-Adapter

Der Dokument-Typ weist dem Dokument eindeutig ein Applikationsmuster und damit einen bestimmten Modellierungskontext zu. Diese Information legt das Regelwerk fest, das für die Transformation des Dokuments in die VMTL verwendet wird. Die Adapter setzen verschiedene Parser ein, je nachdem welches Quellformat die Dokumente aufweisen. Prinzipiell können die Adapter der virtuellen Werkzeugmaschine lediglich Dokumente verarbeiten, die algorithmisch von einem Parser erfasst werden können. Liegen die Dokumente in einer XML-Notation vor, beschränkt sich die Implementierung des Adapters auf die Definition geeigneter XSTL-Regeln, da es für das Einlesen von XML-Daten keines eigens zu implementierenden Parsers bedarf. Die XSLT-Regeln für die Transformation der XML-Daten in die VMTL entsprechen dabei der Interpretationsfunktion I^{ap} aus Definition 4-7. Der Konverter besteht in diesem Fall aus einem XSLT-Prozessor, der als Eingabe-Parameter das XML-Dokument sowie die XSLT-Regeln erwartet. Bei der Transformation selbst handelt es sich vereinfacht um eine Textersetzung, wobei die XML-Tags des Dokuments in korrespondierende Sprachelemente der VMTL übersetzt werden. Für Dokumente, deren Struktur in einem Standard ohne verfügbares XML-Format spezifiziert ist, sind spezielle Parser erforderlich. Bei deren Implementierung können Parser-Generatoren eingesetzt werden. Dabei wird dem Generator die im Standard definierte Dokumentstruktur in Form einer Grammatik als Eingabeparameter übergeben. Der generierte Parser akzeptiert dann alle Dokumente, die den Vorgaben dieser Grammatik entsprechen. Der Konverter ist in diesem Fall eng an die Implementierung des Parsers geknüpft. Die Adapter der virtuellen Werkzeugmaschine werden unabhängig von den Quell- bzw. Zieldokumenten als XML-Adapter bezeichnet, da die Verarbeitung der Daten auf der XML-Notation der VMTL basiert.

6.3 Die Mediator-Schicht

In der Mediator-Schicht werden die zentralen Funktionen implementiert, welche die Integration der fachbereichsspezifischen Daten in den Informationsverbund der virtuellen Werkzeugmaschine ermöglichen. Der *Machine-Tool-Metamodell-Manager* (MTM-Manager) ist hierbei die zentrale Instanz, die alle Informationsflüsse und Kontrollabläufe im Metadaten-Management-System steuert. Er ist als eigenständiges Modul des Applikationsservers ausgelegt. Der MTM-Manager ist innerhalb des Metadaten-Management-Systems für die semantische Integration der Entwicklungsdaten verantwortlich und nutzt die Systemdienstfunktionen der XML-Adapter zum Einlesen der verschiedenen Datenformate. Die XML-Adapter übernehmen somit die syntaktische Integration der Daten. In Abbildung 6-3 sind der Aufbau des MTM-Managers sowie dessen zentrale Funktionen abgebildet.

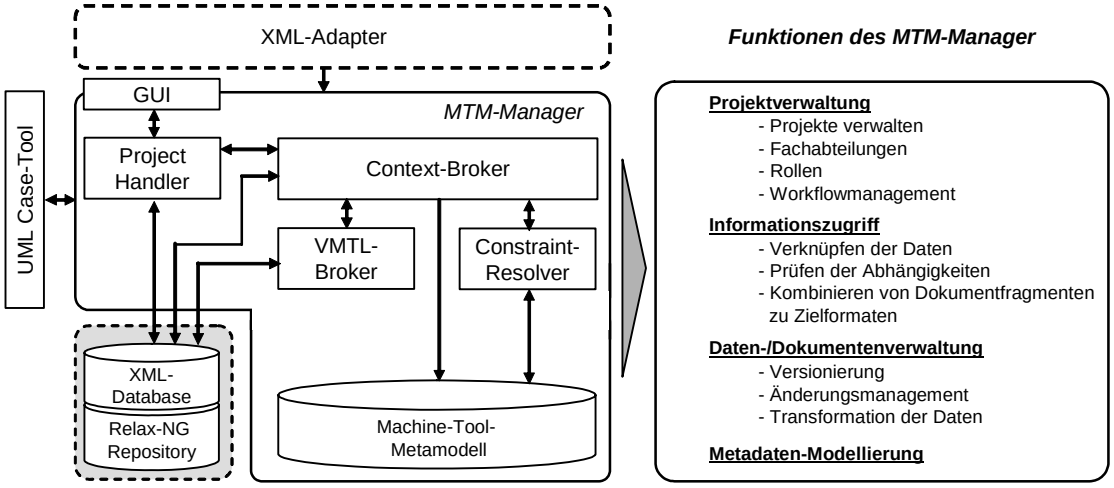


Abbildung 6-3: Datenverarbeitung in der Mediator-Ebene

Der MTM-Manager setzt sich im Wesentlichen aus vier Komponenten zusammen. Der *Project-Handler* ist für die Projektverwaltung zuständig und ermöglicht es, neue Projekte anzulegen und diesen im Rahmen der Workflow-Beschreibung Rollen und Entwicklungsabläufe zuzuweisen. Für die Interaktion mit dem Anwender besitzt der Project-Handler eine GUI-Komponente, über die alle weiteren Funktionen des MTM-Managers ausgeführt werden können. Die Projekte sowie alle Dokumente werden in der XML-Datenbank des Metadaten-Management-Systems verwaltet.

Der *Context-Broker* stellt die Funktionen für die semantische Integration der Entwicklungsdaten zur Verfügung. Zusammen mit dem Project-Manager ermöglicht er den Zugriff auf die in der XML-Datenbank gespeicherten Daten und Dokumente. Insbesondere übernimmt der Context-Broker die Aufgabe der Konsistenzsicherung, indem er direkt auf das Metamodell zugreift. Die in den Dokumenten enthaltenen Datenstrukturen sind im Metamodell abgebildet und mit weiteren Daten aus den einzelnen Fachbereichen über Abhängigkeitsbeziehungen miteinander verbunden. Dieses Netzwerk wird vom Context-Broker algorithmisch durchlaufen, um gezielt Daten auf ihre Vollständigkeit und Widerspruchsfreiheit zu überprüfen. Dabei setzt er den VMTL-Broker ein, der in der XML-Datenbank nach Daten- bzw. Dokumentfragmenten sucht. Die Suchstrategien leiten sich hierbei direkt von den im Metamodell hinter-

legten Abhängigkeitsbeziehungen ab. Darüber hinaus wird durch sie auch festgelegt, welche Dokumentfragmente sich zu einem neuen Dokument zusammenstellen lassen, das anschließend vom XML-Adapter in ein softwarespezifisches Datenformat transformiert werden kann. Die Strukturbeschreibungen der Dokumente liegen in der *Relax NG*-Schemasprache [VLIST 2003] vor und werden von der XML-Datenbank verwaltet.

Bei der Auswertung und Analyse der Daten steht dem Context-Broker des Weiteren der *Constraint-Resolver* zur Verfügung. Im Metamodell können bestimmte Abhängigkeitsbeziehungen durch Constraints näher spezifiziert werden. Constraints sind OCL-Ausdrücke (siehe Anhang 11.2.2), die Bedingungen für die im Metamodell abgebildeten fachbereichsspezifischen Modellkonstrukte und deren Attribute festlegen. Durch geeignete Constraints kann beispielsweise festgelegt werden, dass jeder Baugruppe einer Werkzeugmaschine eine Funktionsangabe zugeordnet werden muss oder aus welchen Teilen sich die Betriebsmittelkennzahl zusammensetzt.

Für die Beschreibung und Definition des Metamodells wird das UML Case-Tool *Poseidon* [GENTLEWARE 2005] als Editor in das Metadaten-Management-System eingebunden. Das Metamodell wird dabei als XMI-Datei abgespeichert und über einen XML-Adapter eingelesen. Die Verwaltung des Metamodells erfolgt im *Metadata Repository* (MDR) des Netbeans-Projekts [MATULA 2005]. Der Editor erlaubt den Aufbau der Klassendiagramme für den Kern und die Applikationsmuster sowie die Spezifikation von Constraints.

6.4 Die Daten-Schicht

Die gesamte Datenverwaltung der virtuellen Werkzeugmaschine erfolgt auf der Grundlage einer XML-Datenbank (siehe Abbildung 6-4). Diese erlaubt das direkte Speichern und Verarbeiten von XML-Daten ohne die Transformation in relationale oder objekt-orientierte Datenmodelle. Der Nutzen, den der Einsatz einer XML-Datenbank im vorliegenden Fall bietet, liegt in der Flexibilität der zu verarbeitenden Dokumente. Der Betrieb einer XML-Datenbank erfordert kein striktes Datenschema, wie dies bei relationalen Datenbanken der Fall ist. Jedes XML-Dokument kann einfach in die Datenbank geladen werden.

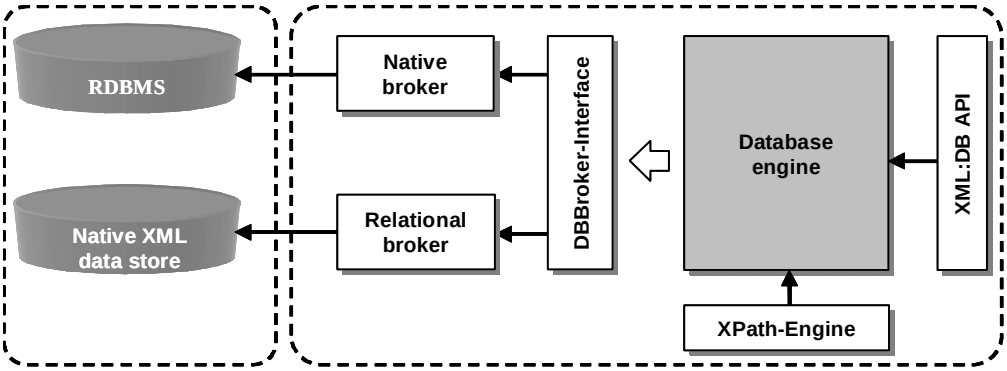


Abbildung 6-4: Verwalten der VMTL-Dokumente mit eXist [CHAUDHRI ET AL. 2003]

Durch XPath (siehe Abschnitt 11.3.1) können die Daten in den einzelnen Dokumenten abgerufen werden. XPath verwendet für die Navigation in XML-Dateien deren baumförmigen

Aufbau und nutzt somit Pfade zum Durchlaufen der Informationen. Die Syntax der Abfragen ähnelt dem Durchsuchen von Dateisystemen. XPath-Abfragen werden von der *XPath-Engine* verarbeitet, die hierfür die Funktionen des *DBBroker* nutzt. Dieser stellt Basis-Operationen zur Verfügung, um Dokumente hinzuzufügen, zu löschen oder entsprechend den XPath-Abfragen zurückzuliefern. Die Schnittstelle des DBBrokers ermöglicht darüber hinaus auch den Zugriff auf eine relationale Datenbank. Diese wird für die Benutzer- und Rechteverwaltung der virtuellen Werkzeugmaschine verwendet. Die Datenbank wird anhand der *xml:db*-API in die Entwicklungsumgebung eingebunden. Die xml:db [XML:DB INITIATIVE 2005] ist eine von Firmen und unabhängigen Entwicklern gegründete Organisation, die sich mit der Erarbeitung von XML-Datenbankstandards beschäftigt. Die Entwicklung einer für alle Datenbanken gültigen API wurde durch die Formulierung der xml:db-API erreicht. Dadurch kann jede beliebige xml:db-konforme Datenbank für die native Verwaltung von XML-Dokumenten verwendet werden. Für das Metadaten-Management-System wurde die XML-Datenbank eXist verwendet [MEIER 2005].

7 Anwendungsbeispiel

7.1 Übersicht

In diesem Kapitel werden zunächst der entwickelte Softwareprototyp erläutert sowie die wichtigsten Komponenten und GUI-Oberflächen vorgestellt. Darüber hinaus werden die Softwarewerkzeuge aufgeführt, die mittels entsprechender Adapter an das Metadaten-Management-System angeschlossen sind. Im Anschluss werden zwei Fallbeispiele beschrieben, welche die Anwendung der Systemmodellierung aus Abschnitt 5.3 anhand eines Modells einer Werkzeugmaschine sowie einer LKW-Bremsanlage aufzeigen.

7.2 Realisierung des Metadaten-Management-Systems

Das Metadaten-Management-System wurde als Softwareprototyp implementiert (siehe Abbildung 4-2) und stellt die Kernkomponente der virtuellen Werkzeugmaschine dar. Es besitzt eine graphische Benutzeroberfläche und eine Reihe von XML-Adaptern für die Anbindung fachspezifischer Softwarewerkzeuge. In den folgenden Abschnitten werden die Funktionen des Softwareprototyps sowie dessen Konfiguration vorgestellt.

7.2.1 Bedienoberfläche

Das Metadaten-Management-System der virtuellen Werkzeugmaschine ist in Kapitel 6 detailliert beschrieben. Die Bedienung des Systems erfolgt durch den Machine-Tool-Metamodell-Manager (MTM-Manager, siehe Abbildung 6-3). Dieser umfasst die grundlegenden Funktionen für das Informationsmanagement sowie für die Projekt-, Daten- und Dokumentenverwaltung. Darüber hinaus stellt er die XMI-Schnittstelle für ein UML Case-Tool zur Verfügung, das für die Modellierung des Machine-Tool-Metamodells verwendet wird.

Die Interaktion mit dem Anwender erfolgt über die Benutzeroberfläche des MTM-Managers. In Abbildung 7-1 a ist die Startoberfläche dargestellt. Über die Menüpunkte sind die einzelnen Funktionen zu erreichen. Unter „Project“ können neue Projekte angelegt, geöffnet oder gelöscht werden. Die Anwendung der einzelnen XML-Adapter für den Im- und Export von Entwicklungsdokumenten wird unter dem Menüpunkt „File“ gesteuert. Unter MTMetamodell wird das externe UML Case-Tool zur Bearbeitung des Metamodells aufgerufen und unter „XML-db“ kann anhand des eXist-Datenbank-Clients die Struktur der Datenbank eingesehen werden. Die Baumansicht zeigt die Struktur des Metamodells mit den einzelnen Applikationsmustern, den dort definierten Sprachkonstrukten sowie dem Metamodell-Kern.

In den Abbildungen 7-1 b und c sind verschiedene Sichten auf die im Projekt enthaltenen Dokumente dargestellt. Die dort abgebildeten Masken werden über den Menüpunkt „Project“ aufgerufen. Links im Bild sind die einzelnen Applikationsmuster als Verzeichnisse zu sehen, in denen die Entwicklungsdokumente als VMTL-Dokumente aufgelistet sind. In der rechten Bildhälfte sind drei Karteireiter abgebildet, welche die einzelnen Dokumente in einer HTML- bzw. VMTL-Ansicht zeigen. In Abbildung 7-1 d ist die dritte Ansicht zu sehen.

Sie zeigt die in Abbildung 5-24 beschriebene graphische Darstellung zur Definition der möglichen Zielformate einer VMTL-Datei.

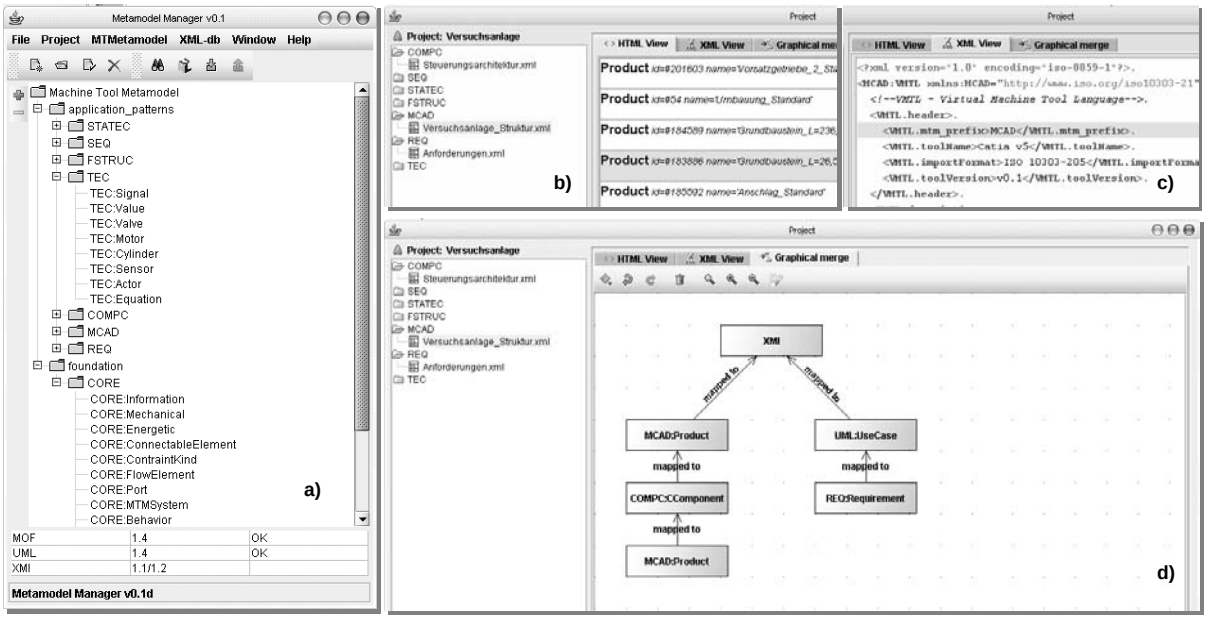


Abbildung 7-1: GUI-Oberfläche des Machine-Tool-Metamodell-Managers

Die Integration von Daten aus verschiedenen VMTL-Dokumenten erfolgt schließlich dadurch, dass in einer weiteren Maske die Quelldokumente, das Zielformat sowie das Applikationsmuster zur Speicherung des Zieldokuments festgelegt werden.

Der implementierte Funktionsumfang des MTM-Managers beinhaltet die grundlegenden Funktionen, um die prinzipielle Anwendbarkeit des vorgeschlagenen Konzepts (siehe Abschnitt 4.5) zu demonstrieren. Die für das Gesamtkonzept der virtuellen Werkzeugmaschine erforderlichen Funktionen zum Dokumenten- bzw. Änderungsmanagement und der damit einhergehenden Verwaltung und Generierung von Betriebsmittelkennzahlen sind in gesonderte Softwarekomponenten zu implementieren.

7.2.2 Realisierte XML-Adapter

Im Folgenden werden die realisierten XML-Adapter des Metadaten-Management-Systems vorgestellt. Diese definieren die Schnittstellen, um die aufgeführten Softwarewerkzeuge zu einem Informationsverbund im Sinne der virtuellen Werkzeugmaschine zu integrieren. Die XML-Adapter konvertieren die von den Softwarewerkzeugen angebotenen Formate in das Zielformat VMTL.

Poseidon

Das UML Case-Tool *Poseidon* der Firma Gentleware wird zur Modellierung und Pflege des Metamodells verwendet und direkt durch die Bedienoberfläche des MTM-Managers aufgerufen.

Das Metamodell wird in Poseidon erstellt bzw. erweitert und anhand der XMI-Schnittstelle direkt in das Metadaten-Management-System importiert.

Open Office

Die Open-Source-Textverarbeitungssoftware *Open Office* dient als einfaches Werkzeug zur Aufnahme und Gliederung von Kunden- und Systemanforderungen. Für die Systemanalyse und die Gliederung der Anforderungen wurden eigene Formatvorlagen erstellt. Dadurch ist es möglich, einzelne Dokumentfragmente mit speziellen Attributen auszuzeichnen und somit Anforderungen zu gliedern und zu gruppieren. Open Office verwendet das XML-basierte *Open Document Format for Office Applications (ODF)* als Dateiformat.

Doors

Das Requirement-Management Werkzeug *Doors* der Firma Telelogic wird für die Anforderungsspezifikation eingesetzt und ist vor allem im Automotive-Bereich weit verbreitet. Doors bietet die XML-basierte Schnittstelle *Requirements Interchange Format (RIF)*, welche die Grundlage für die Implementierung des XML-Adapters darstellt.

Together

Das UML Case-Tool *Together* der Firma Borland wird primär für die Modellierung der Softwarestruktur und zur Verhaltensbeschreibung der Softwarekomponenten eingesetzt. Die erstellten Modelle werden im XMI-Format gespeichert und durch den XML-Adapter als VMTL-Dokument importiert. Der Adapter kann für alle Softwarewerkzeuge verwendet werden, welche das XMI-Format verarbeiten können (z. B. Microsoft Visio).

Pro/ENGINEER und Catia

Die 3D-CAD-Werkzeuge *Pro/ENGINEER* der Firma PTC sowie *Catia* der Firma Dassault Systèmes sind die zentralen Werkzeuge in der Mechanikkonstruktion. Mit diesen Werkzeugen werden die Systemkomponenten modelliert und die Produktstruktur definiert. Ihre Integration in die virtuelle Werkzeugmaschine basiert auf dem Dateiformat ISO 10303-21.

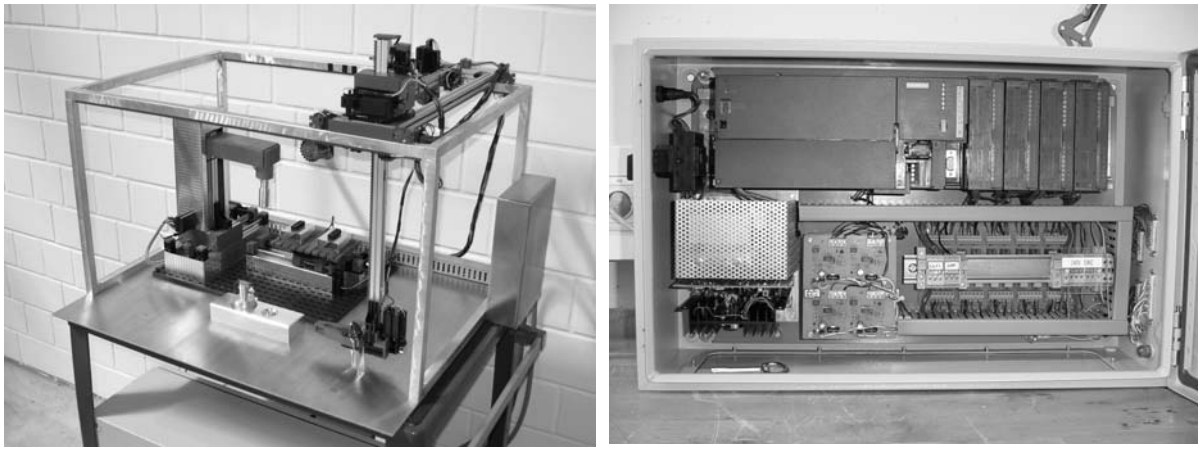
7.3 Fallbeispiel 1: Entwicklung eines Werkzeugmaschinenmodells

Im Rahmen eines studentischen Projekts wurde ein Demonstrator einer Werkzeugmaschine entwickelt und aufgebaut (siehe Abbildung 7-2 a). Die Fachbereiche der Mechanik- und Elektrokonstruktion mit den entsprechenden Aufgabenbereichen wurden durch einzelne Entwicklungsteams nachgebildet. Für die Spezifikation der Anforderungen sowie des Maschinenkonzepts kamen die in dieser Arbeit entwickelten Konzepte zur Systemmodellierung zum Einsatz. Die hierzu verwendeten Softwarewerkzeuge wurden mittels des in Kapitel 6 vorgestellten Metadaten-Management-Systems miteinander gekoppelt. Im Vorfeld des Projekts war es deshalb erforderlich, anhand der in Abschnitt 5.2.2 erläuterten Vorgehensweise ein Metamodell für die Integration der Entwicklungsdaten aufzubauen.

7.3.1 Aufbau der Versuchsanlage

Der Demonstrator stellt ein Fräsbearbeitungszentrum mit drei kartesischen Achsen und vertikaler Hauptspindel dar. Zusätzlich verfügt er über einen Palettenwechsler zum Wechseln der Werkstücke während der Bearbeitung. Des Weiteren ist ein Werkzeugwechselsystem integriert, das als Portalroboter ausgelegt ist. Ein Greifmechanismus ermöglicht das Handhaben der Werkzeuge sowie deren Transport zwischen der Hauptspindel und einem einfachen Leistenmagazin. In einem von der Maschine lokal getrennten Schaltschrank sind eine Leistungsversorgung sowie eine Speicherprogrammierbare Steuerung der Firma Siemens, Modellreihe S7, untergebracht. Um die Komplexität der Umsetzung zu reduzieren, erfolgt die Anbindung der Aktoren nicht über ein standardisiertes Feldbussystem, sondern sie wird durch eine direkte Verkabelung realisiert. Die Ansteuerung der Aktoren ist durch die im Schaltschrank (Abbildung 7-2 b) untergebrachten Relais sichergestellt. Diese sind mit den Ausgängen der Steuerung verknüpft und werden entsprechend der Programmlogik betätigt.

Im Demonstrator kommen zwei Typen von Sensoren zur Anwendung. Einfache Inkrementalgeber dienen zur Lageerfassung, während mechanische Endschalter zur Begrenzung der Verfahrwege benutzt werden. Die Anbindung der Endschalter an die Steuerung wird ebenfalls über eine direkte Verkabelung realisiert. Die zur Lageerfassung notwendige Signalvorverarbeitung in Form einer eigens entwickelten Zählerbaugruppe ist ebenfalls im Schaltschrank untergebracht.



a) Werkzeugmaschine b) Schaltschrank

Abbildung 7-2: Demonstrator eines Fräsbearbeitungszentrums

Durch das Programm des Demonstrators wird ein Fräsbearbeitungsvorgang simuliert. Dabei wird in einem ersten Arbeitsschritt die Oberfläche des Werkstücks plangefräst, anschließend wird das Werkzeug gewechselt und mit der Feinbearbeitung der Werkstückoberfläche der Prozessablauf abgeschlossen. Um eine Störung simulieren und das Verhalten der Maschinensteuerung im Fehlerfall abbilden zu können, ist es möglich, einen Sensor in der Hauptspindel durch einen Schalter zu deaktivieren. Der Sensor meldet an die Steuerung, ob sich ein Werkzeug in der Spindel befindet. Dadurch kann das Wiederanlaufverhalten der Maschine nach Eintreten und Beheben des Fehlers demonstriert werden.

7.3.2 Projektdurchführung

Für die Entwicklung des Werkzeugmaschinendemonstrators wurden zwei Teams gebildet. Ein Team übernahm die Rolle der Mechanikkonstruktion und war vorwiegend für die Anforderungsspezifikation, für die Konzeptentwicklung und für die mechanische Konstruktion sowie für die Fertigung verantwortlich. Dem zweiten Team wurden die Aufgabenbereiche der Elektrokonstruktion übertragen. Es war federführend in der Entwicklung und Ausführung der Aktorik und Sensorik, in der Verkabelung sowie der Programmierung der Steuerungssoftware.

Die Entwicklung des Demonstrators erfolgte nach dem in Abschnitt 5.3 vorgestellten Konzept zur Systemmodellierung. Beide Teams gingen nach der dort erläuterten Vorgehensweise vor und setzten die erläuterten Modellierungstechniken zur Systemspezifikation ein. Die in den einzelnen Entwicklungs- und Konstruktionsschritten eingesetzten Softwarewerkzeuge wurden auf der Basis des Metadaten-Management-Systems miteinander vernetzt. In Abbildung 7-3 sind die verwendeten Softwarewerkzeuge aufgeführt.

Das Lastenheft wurde mit dem frei erhältlichen Textverarbeitungssystem Open Office erstellt. Um die einzelnen Maschineneigenschaften modelltechnisch erfassen zu können, wurden die entsprechenden Textstellen mit den im Applikationsmuster *REQ* (siehe Abschnitt 5.4.3) definierten Klassen von Anforderungen markiert und anhand des XML-Adapters in die Datenbank des Metadaten-Management-Systems geladen. Die Speicherung der Daten erfolgte im eigens entwickelten Zwischenformat VMTL (siehe Abschnitt 5.5), um diese unabhängig von ihrem jeweiligen Quellformat weiter verarbeiten zu können.

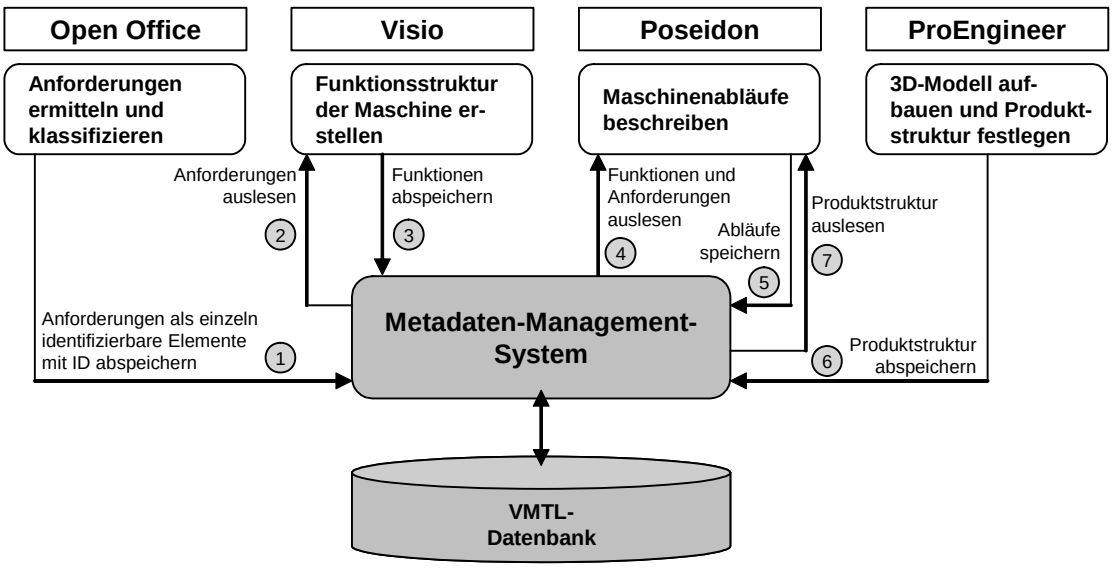


Abbildung 7-3: Einsatz des Metadaten-Management-Systems zur Integration der Entwicklungs- und Konstruktionswerkzeuge

In einem weiteren Schritt bildeten die Projektteams gemeinsam die Funktionsstruktur der Maschine gemäß [PAHL ET AL. 2004], [KOLLER 1985] und [ROTH 2001] mit Microsoft Visio ab und verknüpften diese mit den Anforderungen aus dem Lastenheft. In dem UML Case-Tool Poseidon wurden Zustandsgraphen für die Darstellung der Maschinenabläufe angefertigt und

im Metadaten-Management-System gespeichert. Anschließend wurde mit ProEngineer ein 3D-Modell der Maschine auf der Grundlage der erstellten Funktionsstrukturen und Ablaufbeschreibungen aufgebaut. In einer weiteren Detaillierungsphase wurde die Produktstruktur aus dem 3D-CAD-Modell in das UML Case-Tool eingelesen, die Maschinenabläufe konkretisiert und diese den einzelnen Baugruppen zugeordnet.

Die beiden Projektteams spezifizierten die Maschinenfunktionen sowie deren Struktur ausschließlich mit dem in Abschnitt 5.3 erläuterten Konzept zur Systemmodellierung. Das Metadaten-Management-System sorgte im Hintergrund dafür, dass die in den jeweiligen Softwaresystemen erstellten Modelle in den nachfolgenden Arbeitsschritten als Grundlage für die weitere Feinspezifikation und Modellierung von Maschinenverhalten und -struktur eingesetzt werden konnten. Die aus dem Lastenheft abgeleitete Funktionsstruktur bildete bis zur Inbetriebnahme der Maschine das grundlegende Entwicklungsdokument. Durch die im Metamodell beschriebenen Beziehungen war es möglich, die einzelnen im Verlauf der Entwicklung erstellten Modelle in einen gemeinsamen Entwicklungskontext zu setzen. Dies erlaubte es den Projektteams, auf die für sie erforderlichen Informationen direkt zuzugreifen sowie Rückmeldung darüber zu erhalten, welche Funktionen bzw. Maschineneigenschaften bei Änderungen betroffen waren.

7.4 Fallbeispiel 2: Entwicklung eines LKW-Bremssystems

Die in Abschnitt 5.3 beschriebene Systemmodellierung fokussiert sich auf die Beschreibung der Systemfunktionen einer Werkzeugmaschine. Um die Anwendbarkeit der vorgeschlagenen Vorgehensweise und Modellierungstechniken auf andere mechatronische Systeme zu erweitern, wurde ein Teilbereich eines *Elektronischen Bremssystems* (EBS) für Nutzfahrzeuge modelliert. In den folgenden beiden Abschnitten werden die Motivation sowie die Ergebnisse des Fallbeispiels erläutert.

7.4.1 Aufgabenstellung und Zielsetzung

Die internationale Norm IEC 61508 [DIN EN 61508 2002] stellt eine verbindliche Grundlage für Systeme dar, die eine Sicherheitsfunktion ausführen. Es werden einzelne Prozessschritte beschrieben, die im Sinne der Produkthaftung für die Bremssystemhersteller von zentraler Bedeutung sind. Die Hersteller von Nutzfahrzeugen müssen aufgrund rechtlicher Verbindlichkeiten für die Einhaltung entsprechender Vorgehensweisen und Richtlinien bei ihren Zulieferunternehmen sorgen. In diesem Zusammenhang spielen Reifegradmodelle wie das *Capability Maturity Model Integration* (CMMI) [CMMI 2006] und *Software Process Improvement and Capability Determination* (SPICE) [ISO/IEC 15504-5 2006] eine immer bedeutendere Rolle.

Aufgrund dieser Rahmenbedingungen sind die Bremssystemhersteller angehalten, ihre Entwicklungsmethoden und Modellierungskonzepte der geforderten Prozessqualität anzupassen und zu erweitern.

Im Rahmen des vom *Bayerischen Staatsministerium für Wirtschaft, Verkehr und Technologie* geförderten *Bayerischen Kompetenznetzwerks Mechatronik* (BKM) wurde hierzu im Projekt *Werkzeug zur integrierten Entwicklung von mechatronischen Produktionsmaschinen* ein durchgängiges Modellierungskonzept erarbeitet. Die Zielsetzung des Projektes ist in Abbildung 7-4 dargestellt.

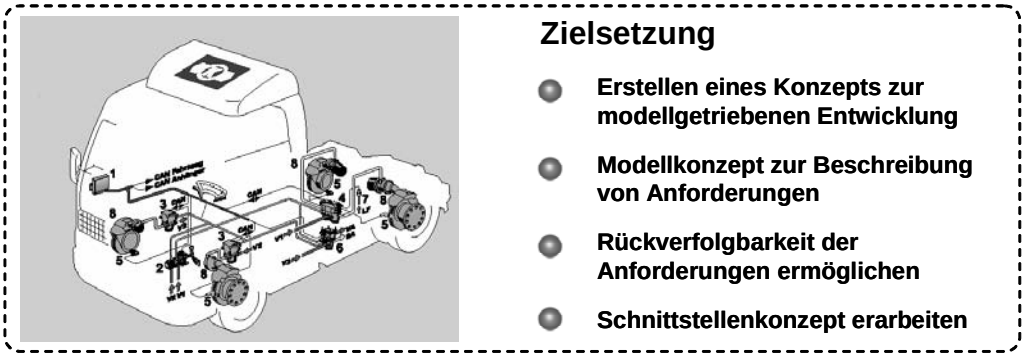
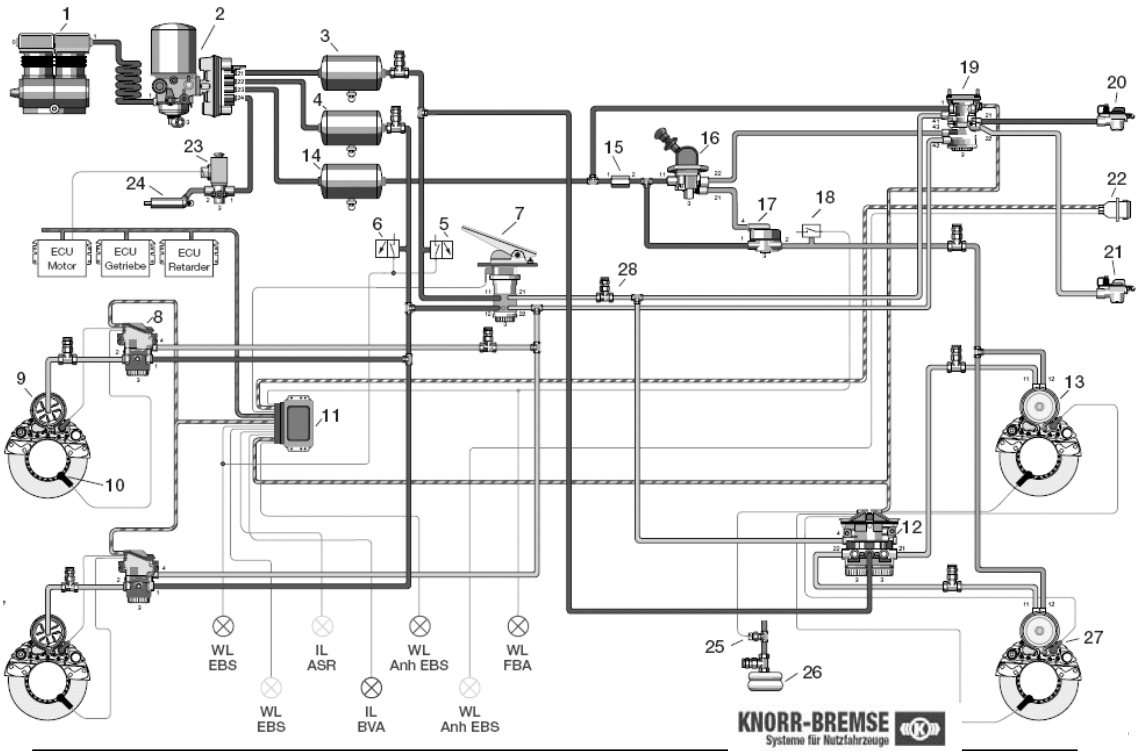


Abbildung 7-4: Ziele des BKM-Projektes zur modellgetriebenen Entwicklung von Bremssystemen für Nutzfahrzeuge

Die Aufgabe bestand darin, die Rückverfolgbarkeit der Anforderungen im Entwicklungsprozess gewährleisten zu können. Voraussetzung hierfür ist eine einheitliche Vorgehensweise und Modellierungstechnik, verbunden mit geeigneten, miteinander vernetzten Softwarewerkzeugen. Die Anforderungsspezifikation für ein elektronisches Bremssystem erfolgte mit standardisierten Checklisten und Fließtext. Als Softwarewerkzeuge wurden herkömmliche Büroapplikationen wie Tabellenkalkulations- und Textverarbeitungssysteme eingesetzt. Das zentrale Dokument für die Spezifikation und Entwicklung ist der so genannte *Bremssystemplan*. Darin werden die zentralen Komponenten sowie deren Funktionsweise im Kontext des Gesamtsystems beschrieben. Der Bremssystemplan ist somit Grundlage für die pneumatische Auslegung ebenso wie für die Entwicklung der Software der Steuergeräte und der Mechanikkonstruktion. In Abbildung 7-5 ist ein Bremssystemplan für eine LKW-Zugmaschine dargestellt.



1 Kompressor	8 EBS-Einkanalmodul	15 Rückschlagventil	22 EBS-Verbindung ISO 7638
2 Luftaufbereitung	9 Membranbremszylinder	16 Handbremsventil	23 Magnetventil
3 Druckluftbehälter Kreis 1	10 Polrad und Drehzahlsensor	17 Relaisventil	24 Motorbremszylinder
4 Druckluftbehälter Kreis 2	11 EBS-Elektron. Steuergerät	18 Druckschalter	25 Drucksensor
5 Druckschalter Kreis 1	12 EBS-Zweikanalmodul	19 EBS-Anhängersteuermodul	26 Luftfederbalg
6 Druckschalter Kreis 2	13 Kombizylinder	20 Kupplungskopf „Vorrat“	27 Verschleißsensierung
7 EBS-Fußbremsmodul	14 Druckluftbehälter Kreis 3	21 Kupplungskopf „Bremse“	28 Prüfanschluss

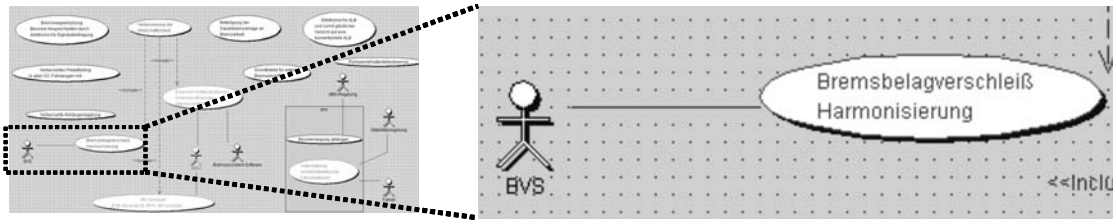
Abbildung 7-5: Bremssystemplan einer Zugmaschine (Knorr-Bremse)

Ein elektronisches Bremssystem vereint die Grundfunktionen einer elektropneumatischen Bremse mit den Funktionen eines *Antiblockiersystems* (ABS) und einer *Antischlupfregelung* (ASR). Im Bremssystemplan sind die Systemkomponenten wie Drehzahlsensoren, Druckregelmodule und Steuergeräte eingezeichnet sowie deren Kommunikation über den CAN-Bus abgebildet. Eine detaillierte Beschreibung der Funktionsweise ist in [DIETSCHE 2007] zu finden. Der Bremssystemplan ist eine einfache 2D-Zeichnung, die aus verschiedenen Vorlagen vorangegangener Projekte zusammengesetzt wird. Für die Erstellung wurde ein herkömmliches Standard-Graphikprogramm eingesetzt, das keine Schnittstellen zu anderen Softwaresystemen besitzt. Um die in Abbildung 7-4 aufgezeigten Zielsetzungen zu erreichen, wurde das in Abschnitt 5.3 vorgestellte Konzept zur Systemmodellierung angewandt. Im folgenden Abschnitt ist die prototypische Umsetzung beschrieben.

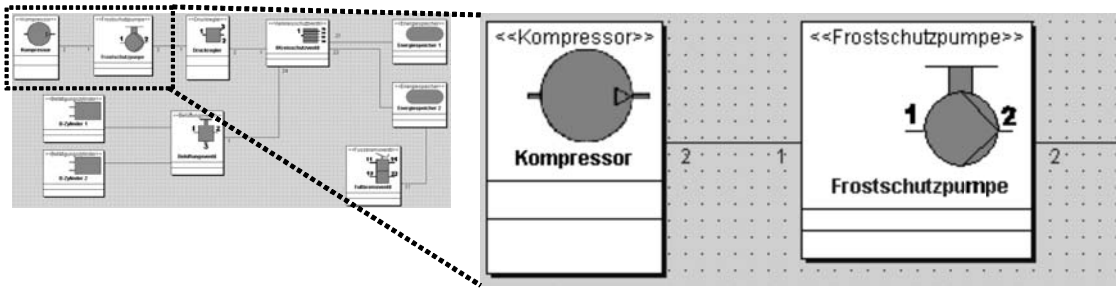
7.4.2 Umsetzung der durchgängigen Prozesskette

Um die Informationen aus der Anforderungsspezifikation für die nachfolgenden Arbeitsschritte im Entwicklungsprozess zur Verfügung zu stellen, wurde eine einheitliche Modellierungstechnik anstelle einzelner, isolierter Entwicklungsdokumente eingeführt. Die UML bildet die Grundlage dieser Technik. Nach einer Analyse der Anforderungsspezifikationen wurde eine Menge an Metadaten identifiziert, die eine Klassifizierung und Einordnung der Anforderungs- und Merkmalsbeschreibungen erlaubt und für alle Entwicklungsprojekte verwen-

det werden kann. Die Anforderungen selbst wurden in Form von Checklisten (*Microsoft Excel*) und Fließtext (*Microsoft Word*) im Requirementsmanagementwerkzeug *Doors* der Firma *Telelogic* in einzelne Textbausteine gegliedert und mit den geeigneten Metadaten annotiert. Über speziell entwickelte XML-Adapter wurden die Anforderungen anschließend in das UML-Werkzeug *Together* der Firma *Borland* importiert und als Use-Case-Diagramm dargestellt (siehe Abbildung 7-6).



a) Requirements als Use-Case-Diagramm



b) Bremssystemplan als erweitertes Klassendiagramm

Abbildung 7-6: Erstellung der Bremssystempläne mit UML

Basierend auf den Anforderungen im Use-Case-Diagramm wurde anschließend in einer iterativen Vorgehensweise der Bremssystemplan erarbeitet. Dieser wird in Form eines UML-Klassendiagramms abgebildet, das um die speziellen Elemente aus der Norm DIN ISO 1219-1 erweitert wurde. Die einzelnen Use Cases können mit dem Bremssystemplan verknüpft werden. Dadurch ist die Rückverfolgbarkeit der Anforderungen vom Bremssystemplan in das Use-Case-Diagramm und über den XML-Adapter nach Doors gewährleistet. Letzteres enthält eine Schnittstelle zu Microsoft-Office-Applikationen, so dass eine bilaterale Verknüpfung zwischen dem Bremssystemplan in Together und der Anforderungsbeschreibung in MS-Word möglich ist.

Durch die exemplarische Umsetzung konnte die praktische Tragfähigkeit der in Abschnitt 5.3 beschriebenen Systemmodellierung sowie der in Abschnitt 6.2 erläuterten XML-Adapter nachgewiesen werden. Insbesondere zeigt die Umsetzung zudem die allgemeine Gültigkeit des entwickelten Ansatzes auch außerhalb des Bereichs der Werkzeugmaschinenentwicklung auf. Im nächsten Kapitel findet sich eine technische und wirtschaftliche Bewertung der in dieser Arbeit vorgestellten Integrationslösung.

8 Technische und wirtschaftliche Bewertung

Der im Entwicklungsprozess vom Maschinenhersteller für die Analyse und Einführung einer Integrationslösung zusätzlich zu erbringende Aufwand lässt sich nur indirekt einem entsprechenden Nutzen gegenüberstellen. Zum einen entziehen sich die Vorteile einer Integrationslösung, wie sie in dieser Arbeit vorgestellt wird, großteils einer Quantifizierung mittels Maßzahlen, die mit den Kosten in Beziehung gebracht werden könnten. Zum anderen hängt der Nutzen im Wesentlichen von den Zielen und der Ausgangssituation im jeweiligen Unternehmen ab. Eine Integrationslösung kann angestrebt werden, um die derzeitigen Entwicklungsabläufe zu optimieren, oder aber um neue Anwendungsgebiete wie neue Simulationsmethoden bzw. -systeme zu erschließen. Während sich die Kosten hierfür relativ genau über die erforderlichen Lizenzen für die Software und den Aufwand bezüglich Analyse, Einführung und Implementierung bestimmen lassen, ist eine entsprechende Analysierung der Vorteile und des Nutzens nur qualitativ möglich.

Management der Entwicklungsinformationen

Nach [FISCHER & VOGEL-HEUSER 2002] wünschen sich viele Anwender bei der Entwicklung automatisierter Systeme einen stärker strukturierten Entwicklungsprozess. Insbesondere sehen sie die Notwendigkeit, die Anforderungsanalyse systematischer durchzuführen, da Fehler in dieser Phase kaum zu identifizieren und zu beheben sind. In [FISCHER &VOGEL-HEUSER 2002] werden beispielhaft die Einsparpotenziale bei der Entwicklung einer Doppelwandanlage beschrieben. Demnach ermöglicht eine systematische Anforderungsanalyse eine Systemspezifikation, welche eine um 10 bis 15 % kostengünstigere technische Lösung, eine Wiederverwendbarkeit von Hard- und Software von 5 bis 15 %, eine Verkürzung der Inbetriebnahmezeit von 20 bis 40 % sowie eine Verkürzung der Durchlaufzeit von 10 bis 20 % ermöglicht. Zunehmend wird in der Automatisierung auch der Einsatz objektorientierter Ansätze diskutiert, um Steuerungssoftware zu erstellen, bei der die Module parametrisiert und angepasst werden können. Diese Möglichkeit soll sich jedoch nicht nur auf die Softwareebene beschränken. Von zunehmender Bedeutung ist die Konfiguration der gesamten Maschine, um die Durchlaufzeiten zu verkürzen und die Entwicklungskosten zu senken. Das Ziel ist es, nach Durchlauf der für die Konfiguration erforderlichen Arbeitsschritte die Spezifikation einer funktionierenden Standardmaschine zu erhalten, ohne zusätzlichen Entwicklungs- und Konstruktionsaufwand zu leisten.

Die Grundlage für dieses Vorhaben sind Modellkonzepte, welche die Maschine mit geeigneten Abstraktionsmitteln beschreiben und insbesondere in der Lage sind, die technischen Abhängigkeiten der einzelnen Baugruppen abzubilden. Die Parametrierung kann dann anhand entsprechender Regelwerke erfolgen, die im Zusammenhang mit der Systemkonfigurierung automatisch Soft- und Hardwarekomponenten vorkonfigurieren. Die Voraussetzung hierfür ist eine wie die in dieser Arbeit vorgestellte Integrationslösung, welche die Daten auf Konsistenz, Widerspruchsfreiheit und Vollständigkeit prüft und somit insgesamt die Datenqualität erhöht. Abbildung 8-1 zeigt zusammenfassend eine Bewertung des in dieser Arbeit vorgestellten Systems zur Integration unterschiedlicher Entwicklungsdaten.

Aspekt	Begründung	Kosten
Datenqualität	Eine zentrale Integrationslösung sorgt dafür, dass die Daten konsistent, widerspruchsfrei und vollständig sind bzw. weist auf entsprechende Fehler in den Daten hin. Damit ist gewährleistet, dass Entscheidungen auf der Basis korrekter Daten gefällt werden.	
Nutzen der bestehenden, fachspezifischen Softwarewerkzeuge	Ein gesamtheitliches Schnittstellenkonzept sorgt dafür, dass Investitionen in teure, fachbereichs-spezifische Softwarelösungen langsfristig erhalten bleiben.	
Langzeitarchivierung	Durch den Einsatz offener Standards im Bereich der Datenformate ist die Transformation der Daten in eine Metasprache wie VMTL möglich. Auch wenn das Originalprogramm nicht mehr verfügbar ist, können die Daten gelesen, geschrieben und mit anderen Softwaresystemen ausgetauscht werden.	
Wiederverwendbarkeit von Daten und Informationen	Informationen liegen nicht nur in einem spezifischen Softwaresystem vor, sondern können auch in anderen Fachbereichen mit den dort verfügbaren Werkzeugen verarbeitet werden. Der Aufwand zur Datensuche und der Abstimmungs-aufwand der Fachbereiche wird minimiert	
Änderungsmanagement	Das Handhaben von Änderungen im Entwicklungsprozess ist komplex und bedeutet vor allem über die Fachbrereiche hinweg eine große Herausforderung. Die Integrationslösung sorgt dafür, dass Änderungen in verschiedenen Modellen und Dokumenten nachgezogen werden.	
Einführung einer Integrations-lösung	Die Einführung ist allgemein mit einem erheblichen Analyse-, Administrations- und Implementierungs-aufwand verbunden. Dieser kann jedoch mit einer methodischen Vorgehensweise (wie in Abschnitt 5.2.2 vorgestellt) gemindert werden.	
Modellierung der fachlichen Abhängigkeiten (Metamodell)	Der Vorteil einer Integrationslösung liegt darin, dass technische und organisatorische Abhängigkeiten zum Großteil automatisiert geprüft werden können. Die Beschreibung dieser Abhängigkeiten bringt jedoch einen hohen initialen Aufwand mit sich.	
Wartung der Integrations-lösung	Für die Anbindung eines jeden neuen Softwaresystems an die Integrationslösung ist das Metamodell zu erweitern sowie entsprechende Adapter zu implementieren. Der Einsatz offener Datenformate reduziert den Analyse- und Implementierungsaufwand.	

Abbildung 8-1: Qualitative Bewertung der Integrationslösung

Das in Abschnitt 4 vorgestellte Konzept der virtuellen Werkzeugmaschine und dabei insbesondere das Metadaten-Management-System als Kern der Integrationslösung sorgen dafür, dass die Entwicklungsdaten in der geeigneten Qualtität vorliegen und im Entwicklungskontext den jeweiligen Fachbereichen zur Verfügung gestellt werden. Die Architektur der

Integrationslösung sieht dabei vor, dass die in den Fachbereichen bereits eingesetzten Softwarewerkzeuge weiter verwendet werden und über spezielle XML-Adapter an das System angebunden sind. Der Aufwand für die Implementierung der Schnittstelle richtet sich nach den zur Verfügung stehenden Ex- und Importformaten bzw. nach der Komplexität der API des jeweiligen Softwarewerkzeugs. In jedem Fall ist nur die Implementierung einer Schnittstelle notwendig, um die Daten zwischen den Softwaresystemen der virtuellen Werkzeugmaschine austauschen zu können. Durch die Digitalisierung von Entwicklungsdaten ist das Thema der Langzeitarchivierung von zunehmender Bedeutung. Ein weiterer wesentlicher Vorteil der vorgestellten Integrationslösung liegt daher darin, dass die Entwicklungsdaten in der herstellerunabhängigen Sprache VMTL abgespeichert werden. Da VMTL eine XML-Sprache ist, kann sie einfach mit herkömmlichen XML-Tools weiterverarbeitet werden, auch wenn die ursprünglichen Softwaresysteme nicht mehr verfügbar sind. Des Weiteren ermöglicht es die virtuelle Werkzeugmaschine, Entwicklungsdaten in anderen Softwarewerkzeugen zu verarbeiten und weiter zu detaillieren. Dadurch sinkt der Aufwand zur Informationssuche sowie zur fachbereichsübergreifenden Abstimmung. Letztere wird vor allem durch das Metamodell unterstützt, das die gegenseitigen Abhängigkeiten der Entwicklungsdaten beschreibt. Somit erhält der Anwender eine Rückmeldung darüber, welche Funktionen und Baugruppen bei der Durchführung von Entwicklungs- und Konstruktionsschritten betroffen sind und gegebenenfalls zusätzlich betrachtet werden müssen. Dieser Aspekt ist unter anderem für das Änderungsmanagement interessant. Das Metamodell beschreibt den Entwicklungskontext der Daten und legt somit die Abhängigkeiten zwischen Funktions-, Komponenten- und Verhaltensbeschreibungen fest. Bei Änderungen kann die Integrationslösung somit Auskunft darüber liefern, welche Teile der Maschine von den Änderungen betroffen und welche Dokumente anzupassen sind.

Schlussfolgerung

Die in dieser Arbeit vorgeschlagene Integrationslösung der virtuellen Werkzeugmaschine stellt eine flexible Möglichkeit dar, Daten aus verschiedenen Entwicklungswerkzeugen zu einem gemeinsamen Entwicklungskontext zusammenzuführen und die Informationssuche zu vereinfachen. Somit ist es möglich, die technische Verzahnung des mechatronischen Systems Werkzeugmaschine auch auf der organisatorischen Ebene der Entwicklung und Konstruktion in geeigneter Form abzubilden. Die virtuelle Werkzeugmaschine leistet darüber hinaus einen Beitrag zum Investitionsschutz, indem bestehende Softwarewerkzeuge und Altsysteme zu einem Informationsverbund integriert und weiter genutzt werden können.

Die Einführung und Wartung der virtuellen Werkzeugmaschine ist jedoch mit einigem Aufwand verbunden. In der Initialisierungsphase ist insbesondere die Modellierung eines stabilen Metamodells zu berücksichtigen, das die fachlichen und technischen Abhängigkeiten der Werkzeugmaschine abbildet. Dies setzt ein fundiertes Wissen der Entwicklungsabläufe und der Systemfunktionen voraus und erfordert mehrere Iterationsschleifen. Fehler bzw. Inkonsistenzen bei der Definition des Metamodells können im laufenden Betrieb nur unter großen Anstrengungen korrigiert werden. Der in diesen Punkten investierte Aufwand amortisiert sich jedoch im laufenden Betrieb durch die in Abbildung 8-1 aufgeführten Kriterien.

9 Zusammenfassung und Ausblick

Den steigenden Marktanforderungen versucht die Werkzeugmaschinenindustrie verstärkt mit neuen Denkansätzen, wie der Mechatronik, zu begegnen. Um weiterhin wettbewerbsfähig zu bleiben, sind moderne, zukunftsorientierte Produkte erforderlich, die sich auf dem Markt behaupten können. Die hierfür notwendige Innovationskraft fokussiert sich dabei zunehmend auf Fachbereiche außerhalb der ursprünglichen Kernkompetenzen der Werkzeugmaschinenhersteller. Die Unternehmen stehen somit vor der Herausforderung, die steigende Komplexität der Maschinen mit neuen Methoden zu beherrschen sowie zusätzliche Vorgehensweisen und Modellierungstechniken für die Entwicklung von Werkzeugmaschinen zu adaptieren.

Dabei versuchen die einzelnen Fachbereiche zunehmend, die Maschineneigenschaften mit geeigneten Modellbeschreibungen direkt am Rechner abzubilden, um die komplexen Wechselwirkungen zwischen den Baugruppen erfassen zu können und in entsprechenden Simulationsumgebungen zu verifizieren. Während Maschinenmodelle im Bereich der Mechanikkonstruktion in Form von MKS- oder FEM-Beschreibungen schon seit längerem eingesetzt werden, ist diese Vorgehensweise noch nicht fachbereichsübergreifend umgesetzt. Beispielsweise werden bei der Entwicklung der Software für speicherprogrammierbare Steuerungen (SPS) Modellierungstechniken, wie beispielsweise Zustandsgraphen, nicht direkt zur Codierung der Steuerungsfunktionalität verwendet. Hier kommen Modellbeschreibungen häufig nur für die nachträgliche Dokumentation der Steuerungslogik zum Einsatz. Das Ziel der Unternehmen ist es deshalb, durch einen fachbereichsübergreifenden modellgetriebenen Ansatz ihre Entwicklungsprozesse zu optimieren, um effizienter, flexibler und kostengünstiger auf die Marktanforderungen reagieren zu können.

Der Begriff *Modellgetrieben* bedeutet in diesem Zusammenhang, dass Anforderungsdokumente, CAD-Zeichnungen, Symboltabellen, E/A-Listen, Funktionspläne und Softwarecode zusammen genommen alle relevanten Aspekte der Maschinenentwicklung abdecken, schlüssig und konsistent aufeinander aufbauen und insbesondere zu jedem Zeitpunkt den aktuellen Entwicklungsstand widerspiegeln. Während die modellgetriebene Vorgehensweise bei der Entwicklung von Businesssoftware und eingebetteter Systeme zunehmend Akzeptanz findet, spielt dieser Ansatz in der Werkzeugmaschinenindustrie noch eine geringe Rolle. Dies ergibt sich daraus, dass die einzelnen Fachbereiche zum Teil völlig unterschiedliche Methoden, Modellierungstechniken und Begrifflichkeiten verwenden und ihren Aufgabenbereich mit untereinander inhomogenen Strukturierungsmitteln bearbeiten. Des Weiteren existieren mittlerweile zwar zahlreiche Modellierungstechniken für die einzelnen Fachbereiche, diese sind derzeit aber meist als Insellösungen ohne direkten Bezug zu den anderen Disziplinen umgesetzt.

Für die Einführung einer modellgetriebenen Entwicklung im Werkzeugmaschinenbau sind deshalb zwei Aspekte wesentlich. Erstens müssen für alle relevanten Fachbereiche geeignete Modellierungssprachen bzw. -techniken vorhanden sein und zweitens ist ein zentrales Softwaresystem notwendig, das die einzelnen Modelle in den Gesamtkontext der Maschinenentwicklung einbettet. Eine derartige Integrationslösung hat im Wesentlichen die Funktionen eines herkömmlichen PDM-Systems, geht aber über die reine Verwaltung, Versionierung und Beschreibung der eingecheckten Dokumente durch Attribute weit hinaus. Zudem müssen die einzelnen Modelle vom System interpretiert, das heißt analysiert und mit weiteren Modellen verglichen werden, um fachliche Abhängigkeiten prüfen und verifizieren zu können. Zusätzlich muss es möglich sein, aus den verwalteten Modellen einzelne Daten und Informationen zu neuen Modellbeschreibungen zusammenzufassen und den entsprechenden Fachbereichen zur Verfügung zu stellen. Herkömmliche PDM-Systeme bieten diese Funktionalitäten zum

Teil lediglich im Bereich der Mechanikkonstruktion an und eignen sich aus diesem Grund derzeit nicht als Plattform für eine modellgetriebene Entwicklungsumgebung.

Abgeleitet aus diesen Erkenntnissen wurde in dieser Arbeit die virtuelle Werkzeugmaschine als eine Softwareplattform für die modellgetriebene Entwicklung im Werkzeugmaschinenbau präsentiert. Sie dient als Integrationslösung für die bei der Maschinenentwicklung eingesetzten Softwareanwendungen und Modellierungstechniken. Der Fokus der Arbeit liegt auf der Kernkomponente der virtuellen Werkzeugmaschine, dem Metadaten-Management-System. Es fungiert als Mediator zwischen den fachbereichsspezifischen Modellkonzepten, indem es in einem Metamodell-Kern die zentralen Elemente der bei der Werkzeugmaschinenentwicklung eingesetzten Modellierungstechniken formal und rechnerinterpretierbar abbildet. Einzelne Softwareanwendungen werden in die virtuelle Werkzeugmaschine integriert, indem so genannte Applikationsmuster definiert und dem Metamodell hinzugefügt werden. Sie leiten sich direkt von dem Metamodell-Kern ab und definieren spezifische, in einem Fachbereich eingesetzte Modellbeschreibungsformen. Der Metamodell-Kern und das Applikationsmuster ergeben zusammen den Gesamtkontext der Maschinenentwicklung. Das Metamodell setzt die in den einzelnen Fachbereichen eingesetzten Modelle in Bezug zueinander und bildet somit deren fachbereichsübergreifende Semantik ab. Weitere Funktionen der virtuellen Werkzeugmaschine, wie das Änderungsmanagement, Konsistenzchecks oder die Generierung von Entwicklungsdokumenten, basieren letztendlich alle auf dem im Metamodell formal hinterlegten Prozesswissen.

Für die Anbindung der einzelnen Softwareanwendungen an die virtuelle Werkzeugmaschine wurden so genannte XML-Adapter entwickelt. Diese importieren die Entwicklungsdaten und konvertieren diese in die eigens entwickelte Virtual Machine Tool Language (VMTL). Die VMTL ist eine XML-Sprache, die sich direkt aus den im Metamodell definierten Modellelementen ableitet und zur Speicherung der fachbereichsübergreifend relevanten Entwicklungsdaten innerhalb der virtuellen Werkzeugmaschine dient.

Das Konzept der virtuellen Werkzeugmaschine wurde als Softwareprototyp umgesetzt und bei der Entwicklung eines Demonstrator-Fräsbearbeitungszentrums sowie eines LKW-Bremssystems eingesetzt. In beiden Anwendungsszenarien konnte die Tragfähigkeit der erarbeiteten Lösungen nachgewiesen und deren Vorteile bei der Datenintegration zwischen verschiedenen Softwareanwendungen belegt werden.

Potenziale für weiterführende Arbeiten bieten die Umsetzung zusätzlicher STEP-Applikationsprotokolle in Bezug auf die Implementierung von XML-Adaptern. Insbesondere die Unterstützung der Serien ISO 10303-28 „STEPml“ sowie ISO 10303-25 „Express to OMG binding“ verspricht zusätzliches Potenzial für die Integration weiterer Modellierungstechniken. Ebenso ist der weitere Ausbau von PDM-Funktionalitäten wünschenswert, wobei hier die Umsetzung eines Workflow- und Versionierungssystems zu priorisieren ist. Das Metamodell wird derzeit durch Klassendiagramme beschrieben und vom Metadaten-Management-System eingelesen. Um das Prozesswissen, das in diesen Diagrammen abgebildet ist, präziser beschreiben zu können, ist der Einsatz einer formalen Sprache wie beispielsweise der OCL entscheidend. Damit wäre es möglich, die Semantik der einzelnen Modellelemente genauer zu definieren und deren fachliche Abhängigkeiten zu detaillieren. Dies hätte beispielsweise nicht nur für das Änderungsmanagement Vorteile, da somit die Auswirkungen auf verschiedene Baugruppen der Maschine genauer abgebildet werden könnten, sondern auch für die Implementierung leistungsfähiger Generatoren, die bei der Erstellung neuer Mo-

dellbeschreibungen eingesetzt werden. Die Logik hierfür ist derzeit noch fest im Softwarecode des Metadaten-Management-Systems implementiert. Für eine flexible Lösung sind jedoch parametrierbare und konfigurierbare Generatoren notwendig, um die von ihnen generierten Modelle und Dokumente direkt für weitere Entwicklungsaufgaben einsetzen zu können. Die vorliegende Arbeit ist also ein wesentlicher Meilenstein auf dem Weg zu einem branchenweiten Einsatz der virtuellen Werkzeugmaschine. Gleichwohl sind im Hinblick auf deren standardmäßige Nutzung noch viele weitere Schritte zu vollziehen.

10 Literatur

Ahle 2000

Ahle, U.: Bedeutung von PDM-Systemen für die virtuelle Produktentstehung - Aktuelle Marktentwicklungen und Trends. In: Krause, F.-L.; Tang, T.; Ahle, U. (Hrsg.): Proceedings Innovationsforum Virtuelle Produktentstehung, Berlin. Berlin: Eigenverlag 2000, S. 337-349.

Ahrens 2000

Ahrens, G.: Das Erfassen und Handhaben von Produktanforderungen. Berlin: Technische Universität Berlin 2000.

Albertz 1995

Albertz, F.: Dynamikgerechter Entwurf von Werkzeugmaschinen-Gestellstrukturen. Berlin: Springer 1995. (*iwb* Forschungsberichte 93).

Albrecht & Meyer 2002

Albrecht, H.; Meyer, D.: XML in der Automatisierungstechnik - Babylon des Informationsaustausches? at - Automatisierungstechnik 50 (2002) 2, S. 87-96.

Anderl et al. 2002

Anderl, R.; Kleiner, S.; Krastel, M.: Produktdatenmanagement in der Simulation und Berechnung. Produktdaten Journal 9 (2002) 105, S. 37-41.

Anton et al. 2002

Anton, O.; Lercher, B.; Reinhart, G.: Modelling of Faults and Fault Recovery: An Essential Aspect of Mechatronic System Design. In: Annals of 2002 Int'l CIRP Design Seminar, Hong Kong. 2002.

Arnold et al. 2005

Arnold, V.; Dettmering, H.; Engel, T.; Karcher, A.: Product Lifecycle Management beherrschen. Berlin: Springer 2005.

Aßmann 1996

Aßmann, S.: Methoden und Hilfsmittel zur abteilungsübergreifenden Projektierung komplexer Maschinen und Anlagen. Karlsruhe: Shaker 1996.
(WZL Aachen - Berichte aus der Produktionstechnik 13).

Balzert 1999

Balzert, H.: Lehrbuch der Objektmodellierung. Heidelberg: Spektrum 1999.

Balzert 1996

Balzert, H.: Lehrbuch der Softwaretechnik. Heidelberg: Spektrum 1996.

Baudisch 2003

Baudisch, T.: Simulationsumgebung zur Auslegung der Bewegungsdynamik des mechatronischen Systems Werkzeugmaschine. München: Utz 2003.
(*iwb* Forschungsberichte 179).

Bender 1999

Bender, K.: Herausforderungen in der Entwicklung eingebetteter Systeme. In: Vortrag im Rahmen der vordringlichen Aktion "Entwicklung, Produktion und Service von Software für eingebettete Systeme in der industriellen Produktion" am 29.11.1999.

Berners-Lee 1997

Berners-Lee, T.: Metadata Architecture. http://www.w3.org/DesignIssues/Metadata.html, 1997

Berners-Lee et al. 2001

Berners-Lee, T.; Hendler, J.; LAssila, O.: The Semantic Web.
http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html: 2001.

Bernstein & Dayal 1994

Bernstein, P. A.; Dayal, U.: An Overview of Repository Technology. In: VLDB 94 (Hrsg.): Proceedings of 20th International Conference on Very Large Data Base. Santiago de Chile: 1994, S. 705-713.

Bernstein 1998

Bernstein, P. A.: Repositories and object oriented databases. SIGMOD Record 27 (1998) 1, S. 8-96.

Birli et al. 1997

Birli, O.; Kallenbach, E.; Saffert, E.; Schäffle, C.: Zur Gestaltung integrierter mechatronischer Produkte. In: Mechatronik im Maschinenbau. Düsseldorf: VDI-Verlag 1997
(VDI Berichte 1315).

Bludau & Welp 2005

Bludau, C.; Welp, E. G.: Semantic Web Services für den wissensbasierten Entwurf mechatronischer Systeme. In: Gausemeier, J. (Hrsg.): Intelligente mechatronische Systeme, Paderborn. Paderborn: Bonifatius GmbH 2005, S. 209-220.
(HNI-Verlagsschriftenreihe 163).

Born et al. 2004

Born, M.; Holz, E.; Kath, O.: Softwareentwicklung mit UML 2 - Die neuen Entwurfstechniken UML 2, MOF 2 und MDA. München: Addison-Wesley 2004.

Box et al. 2001

Box, D.; Skonnard, A.; Lam, J.: Essential XML - XML für die Softwareentwicklung. Boston: Addison-Wesley 2001.

Brockhaus 2006

Brockhaus F. A.: Brockhaus - Die Enzyklopädie in 30 Bänden. 21. Aufl. Mannheim: F. A. Brockhaus 2006.

Bry et al. 2001

Bry, F.; Kraus, M.; Olteanu, D.; Schaffert, S.: Semistrukturierte Daten. Informatik Spektrum 24 (2001) 4, S. 230-233.

Buchner et al. 2000

Buchner, H.; Rauprich, G.; Ahrens, W.: Was bringt die XML der Prozessleittechnik - Buzzword oder informationstechnischer Backbone für eCommerce, Planung und Betriebsbetreuung?. atp - Automatisierungstechnische Praxis 42 (2000) 9, S. 51-62.

Buneman et al. 1997

Buneman, P.; Davidson, S.; Fernandez, M.; Suciu, D.: Adding Structure to Unstructured Data. In: 6th International Conference on Database Theory (Hrsg.): Lecture Notes in Computer Science. Berlin: Springer 1997, S. 336-350.

Böger 1998

Böger, F. H.: Herstellerübergreifende Konfigurierung modularer Werkzeugmaschinen. Düsseldorf: VDI 1998. (Berichte aus dem IFW 462).

Bühler 1965

Bühler, K.: Die Darstellungsform der Sprache. Jena: G. Fischer 1965.

Campbell et al. 2003

Campbell, C. E.; Eisenberg, A.; Melton, J.: XML Schema. SIGMOD Record 32 (2003) 2, S. 96-101.

Cardelli 2001

Cardelli, L.: Describing Semistructured Data. SIGMOD Record 30 (2001) 4, S. 80-85.

Chaudhri et al. 2003

Chaudhri, A. B.; Rashid, A.; Zicari, R.: XML Data Management. Boston: Addison-Wesley 2003.

Chouikha et al. 1998

Chouikha, M.; Janhsen, A.; Schieder, E.: Klassifikation und Bewertung von Beschreibungsmitteln für die Automatisierungstechnik. at - Automatisierungstechnik 46 (1998) 12, S. 582-591.

Clark & DeRose 1999

Clark, J.; DeRose, S.: XML Path Language (XPath) Version, http://www.w3.org/TR/xpath, 1999.

Cluet et al. 1998

Cluet, S.; Delobel, C.; Simeon, J.; Smaga, K.: Your mediators need data conversion! In: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data (Hrsg.): International Conference on Management of Data and Symposium on Principles of Database Systems. New York: ACM Press 1998, S. 177-188.

CMMI 2006

Carnegie Mellon Software Engineering Institute: CMMI for Development Version 1.2, http://www.sei.cmu.edu/pub/documents/06.reports/pdf/06tr008.pdf, 13.08.2006.

Codd 1970

Codd, E. F.: A Relational Model for Shared Databanks. Communications of the ACM 13 (1970) 6, S. 377-387.

Conrad 1997

Conrad, S.: Föderierte Datenbanksysteme, Konzepte der Datenintegration. Berlin: Springer 1997.

Daconta et al. 2003

Daconta, M. C.; Obrst, L. J.; Smith, K. T.: The Semantic Web: A Guide to the Future of XML, Web Services and Knowledge Management. New York: Wiley 2003.

Dahlkvist 2001

Dahlkvist Persson, A.: Product Data Management and Software Configuration Management - Similarities and Differences, http://citeseer.ist.psu.edu/dahlqvist01product.html, 05.08.2005.

DCMI 2004

Dublin Core Metadata Initiative: Dublin Core Metadata Element Set, Version 1.1, Reference Description, http://dublincore.org/documents/2004/12/20/dces/, 07.05.2005.

DCMI 2006

Dublin Core Metadata Initiative: Dublin Core Metadata Initiative, http://dublincore.org, 04.03.2006.

Deifel 2001

Deifel, B.: Requirements Engineering komplexer Standardsoftware. München: Dissertation TU München 2001.

Denninger & Peters 2004

Denninger, S.; Peters, I.: Enterprise JavaBeans. München: Addison-Wesley 2000.

Dietsche 2007

Dietsche, K. H.: Kraftfahrtechnisches Taschenbuch. Wiesbaden: Vieweg 2007.

DIN 199 1977

DIN 199: Begriffe im Zeichnungs- und Stücklistenwesen. Berlin: Beuth 1977.

DIN 6580 1985

DIN 6580: Begriffe der Zerspantechnik; Bewegungen und Geometrie des Zerspanvorganges. Berlin: Beuth 1985.

DIN 6779 1992

DIN 6779: Kennzeichnungssystematik für technische Produkte und technische Produktdokumentation. Berlin: Beuth 1992.

DIN 19226 1984

DIN 19226: Teil 1: Regelungs- und Steuerungstechnik: Begriffe, Allgemeine Grundlagen. Berlin: Beuth 1984.

DIN 66025 1981

DIN 66025: Programmaufbau für numerisch gesteuerte Arbeitsmaschinen. Berlin: Beuth 1981.

DIN EN 60617 1997

DIN EN 60617: Graphische Symbole für Schaltpläne. Berlin: Beuth 1997.

DIN EN 61082 1996

DIN EN 61082: Dokumente der Elektrotechnik. Berlin: Beuth 1996.

DIN EN 61508 2002

DIN 61508: Funktionale Sicherheit sicherheitsbezogener elektrischer/elektronischer/programmierbarer elektronischer Systeme. Berlin: Beuth 2002.

DIN ISO 1219 2004

DIN ISO 1219: Fluidtechnik - Graphische Symbole und Schaltpläne. Berlin: Beuth 2004.

DIN ISO 10007 2004

DIN ISO 10007: Qualitätsmanagement - Leitfaden für Konfigurationsmanagement. Berlin: Beuth 2004.

Doan & Halevy 2005

Doan, A.; Halevy, A. Y.: Semantic Integration Research in the Database Community: A Brief Survey. AI Magazine 26 (2005) 1, S. 83-94.

Dürrschmidt 2001

Dürrschmidt, S.: Planung und Betrieb wandlungsfähiger Logistiksysteme in der variantenreichen Serienproduktion. München: Utz 2001. (*iwb* Forschungsberichte 152).

Ehrenstrasser et al. 2003

Ehrenstrasser, M.; Pörnbacher, C.; Wünsch, G.: Hardware-in-the-Loop-Simulation für die Vorabinbetriebnahme von Steuerungssoftware. In: Zäh, M. F.; Reinhart, G. (Hrsg.): Mechatronische Produktionssysteme - Die virtuelle Werkzeugmaschine, Garching. München: Herbert Utz 2003, S. 7.1-7.17. (*iwb* Seminarberichte 66).

Ehrlenspiel 1995

Ehrlenspiel, K.: Integrierte Produktentwicklung. Mpnchen: Carl Hanser 1995.

Ehrler et al. 1999

Ehrler, A.; Kunze, H.; Maier, M.: Development of Configurable Semantic Mappers. In: PDTAG-AM Group (Hrsg.): Proceedings of the European Conference Product Data Technology Days 1999, Stavanger, Norway. 1999, S. 247-254.

Eigner & Stelzer 2001

Eigner, M.; Stelzer, R.: Produktdatenmanagment-Systeme. Berlin: Springer 2001.

Elmqvist 1978

Elmqvist, H.: A Structured Model Language for Large Continuous Systems. Lund: 1978.

Englberger et al. 2003

Englberger, G.; Guserle, R.; Lercher, B.; Pörnbacher, C.: Die Virtuelle Werkzeugmaschine als integrierter Entwicklungsarbeitsplatz. In: Zäh, M. F.; Reinhart, G. (Hrsg.): Mechatronische Produktionssysteme - Die Virtuelle Werkzeugmaschine, Garching. München: Herbert Utz 2003, S. 0-28. (*iwb* Seminarberichte 66).

Fallside & Walmsley 2004

Fallside, D. C.; Walmsley, P.: XML Schema Part 0: Primer Second Edition, http://www.w3.org/TR/2004/REC-xmlschema-0-20041028, 05.08.2005.

Feldvoss 2003

Feldvoss, B.: Product Data Exchange inside Airbus and within Supply Chain. In: Frick, C.; Katzenbach, A.; Krause, F.-L. (Hrsg.): Proceedings Advances in Collaborative Product Creation, Dresden. Stuttgart: Fraunhofer IRB Verlag 2003, S. 27-35.

Finke 1977

Finke, R.: Berechnung des dynamischen Verhaltens von Werkzeugmaschinen. Dissertation RWTH Aachen 1977.

Fischer & Vogel-Heuser 2002

Fischer, K.; Vogel-Heuser, B.: UML in der automatisierungstechnischen Anwendung - Stärken und Schwächen. atp 44 (2002) 10, S. 63-68.

Flath 2002

Flath, M.: Methode zur Konzipierung mechatronischer Produkte. Paderborn: Bonifatius GmbH 2002. (HNI-Verlagsschriftenreihe 108).

Fleckenstein 1987

Fleckenstein, J.: Zustandsgraphen für SPS - Grafikunterstützte Programmierung und steuerungsunabhängige Darstellung. Berlin: Springer 1987. (ISW Forschung und Praxis 63).

Föderal 2004

N. N.: Abschlussbericht: Baukastenorientiertes Engineering mit Föderal - Ein Leitfaden für Maschinen- und Anlagenbauer. Frankfurt am Main: VDMA-Verlag 2004.

Fritzson 2004

Fritzson, P.: Principles of Object-Oriented modeling and Simulation with Modelica 2.1. New York: Wiley-Interscience 2004.

Garcia-Molina et al. 2000

Garcia-Molina, H.; Ullmann J. D.; Widom, J.: Database System Implementation. New Jersey: Prentice Hall 2000.

Geisberger 2005

Geisberger, E.: Requirements Engineering eingebetteter Systeme - ein interdisziplinärer Modellierungsansatz. Karlsruhe: Shaker 2005.

Genersereth & Fikes 1992

Genesereth, M. R.; Fikes, R. E.: Knowledge Interchange Format, http://www-ksl.stanford.edu/knowledge-sharing/kif/, 06.06.2005.

Gentleware 2005

Gentleware: Poseidon for UML, http://www.gentleware.com/, 06.08.2005.

Gerhard 2000

Gerhard, D.: Erweiterung der PDM-Technologie zur Unterstützung verteilter kooperativer Produktentwicklungsprozesse. Aachen: Shaker 2000. (Schriftenreihe Institut für Konstruktionstechnik 3).

Glatz 2000

Glatz, R.: Software - Wertschöpfungsfaktor in der Produktion. 2000.

Grabowski et al. 2003

Grabowski, H.; Lossack, R.; Hornberg, O.; Ehrler, A.: An Integrated Framework to Support Cross-Cultural Engineering Collaboration. In: Frick, C.; Katzenbach, A.; Krause, F.-L. (Hrsg.): Proceedings Advances in Collaborative Product Creation, Dresden. Stuttgart: Fraunhofer IRB Verlag 2003.

Großmann & Arndt 2000

Großmann, K.; Arndt, H.: Simulation lagegeregelter Vorschubantriebe. In: Großmann, K.; Wiemer, H. (Hrsg.): Dresdner Tagung Simulation im Maschinenbau, Dresden. Dresden: Eigenverlag 2000, S. 461-476.

Großmann 2002

Großmann, K.: Was ist, soll und kann die virtuelle Werkzeugmaschine?. In: Was kann die virtuelle Werkzeugmaschine? In: 4. Dresdner WZM-Fachseminar: Was kann die virtuelle Werkzeugmaschine? Dresden. Tagungsband 2002.

Gruber 1993

Gruber, T. R.: A translation approach to portable ontology specifications. Knowledge Acquisition 5 (1993) 2, S. 199-220.

Gruninger & Lee 2002

Gruninger, M.; Lee, J.: Ontology - Applications and Design. Communications of the ACM 45 (2002) 2, S. 39-41.

Gräb 2001

Gräb, R.: Parametrische Integration von Produktmodellen für die Entwicklung mechatronischer Produkte. Darmstadt: Shaker 2001. (Forschungsberichte aus dem Fachgebiet Datenverarbeitung in der Konstruktion 17).

Haman & Tröndle 1997

Haman, J.; Tröndle, H. P.: Die Schallgrenze bei der Regelung elastomechanischer Systeme. In: Kasper, R. u. a. (Hrsg.): Entwicklungsmethoden und Entwicklungsprozesse im Maschinenbau, Magdeburg. Berlin: Logos 1997, S. 193-203.

Harel 1987

Harel, D.: Statecharts: A Visual Formalism for Complex Systems. Science of Computer Programming 8 (1987), S. 231-274.

Harold 2004

Harold, E. R.: XML - Das mitp-Standardwerk zur professionellen Programmierung mit XML. Bonn: mitp 2004.

Harold & Means 2004

Harold, E. R.; Means, W. S.: XML in a Nutshell. Beijing: O Reily 2004.

Haug 1989

Haug, E. J.: Computer Aided Kinematica and Dynamics of Mechanical Systems. Boston: Allyn and Bacon 1989.

Hengel 1993

Hengel, K.: Softwareentwicklung für speicherprogrammierbare Steuerungen im integrierten, rechnergestützen Konstruktionsprozess. Berlin: Springer 1993. (IPA-IAO Forschung und Praxis 190).

Hesse 2002

Hesse, W.: Ontologie(n). Informatik Spektrum 25 (2002) 6, S. 477-480.

Heverhagen 2003

Heverhagen, T.: Integration von Sprachen für speicherprogrammierbare Steuerungen in die Unified Modeling Language durch Funktionsbausteinadapter. Karlsruhe: Shaker 2003.

Hippmann 2004

Hippmann, G.: Modellierung von Kontakten komplex geformter Körper in der Mehrkörperdynamik. 2004. (Institut für Mechanik und Mechatronik DLR-IB 515-04-35).

Hoffmann 1998

Hoffmann, J.: Matlab und Simulink: Beispielorientierte Einführung in die Simulation dynamischer Systeme. Bonn: Addison-Wesley-Longman 1998.

Hopcroft & Ullmann 1994

Hopcroft, J. E.; Ullmann, J. D.: Einführung in die Automatentheorie, formale Sprachen und Komplexitätstheorie. Bonn: Addison-Wesley 1994.

Huang 2002

Huang, M.: Funktionsmodellierung und Lösungsfindung mechtronischer Produkte. Karlsruhe: Shaker 2002. (Forschungsberichte RPK 2).

IBM 2005

IBM: IBM Lotus Notes, http://www-306.ibm.com/software/de/lotus/, 08.06.2005.

ISO 10303-11 2004

ISO 10303-11: Industrielle Automatisierungssysteme und Integration - Produktdarstellung und -austausch - Teil 11: Beschreibungsmethoden: Handbuch der Modellierungssprache EXPRESS. Berlin: Beuth 2004.

ISO 10303-21 2002

ISO 10303-21: Industrial automation systems and integration - Product data representation and exchange - Part 21: Implementation methods: Clear text encoding of the exchange structure. Berlin: Beuth 2002.

ISO 10303-28 2003

ISO 10303-28: Industrielle Automatisierungssysteme und Integration - Produktdatendarstellung und -austausch - Teil 28: Implementierungsmethoden: XML Repräsentation von EXPRESS-Schemata. Berlin: Beuth 2003.

ISO 10303-203 2004

ISO 10303-203: Application Protocol: Configuration controlled 3D designs of mechanical parts and assemblies. Technical Corrigendum 3. Berlin: Beuth 2004.

ISO 10303-212 2004

ISO 10303-212: Industrielle Automatisierungssysteme und Integration - Produktdatendarstellung und -austausch - Teil 212: Anwendungsprotokoll: Elektrische/elektrotechnische Systeme und Anlagen. Berlin: Beuth 2004.

ISO DIS 23570 2005

ISO DIS 23570: Industrial automation systems and integration - Distributed installation in industrial applications. Berlin: Beuth 2005.

ISO/IEC DIS 26300 2006

ISO IEC DIS 26300: Open Document Format for Office Applications (OpenDocument) v1.0, http://www.iso.org, 06.06.2006.

ISO/IEC JTC 1/SC 34 2002

ISO/IEC JTC 1/SC 34: Document Schema Definition Languages (DSDL) - Part 2: Regular-grammar-based validation - RELAX NG, http://www.y12.doe.dov/sgml/sc34/document/0362_files/relaxng-is.pdf: 2002.

ISO/IEC 15504-5 2006

ISO IEC 15505-5:2006: Informationstechnik - Prozessbewertung - Exemplarisches Prozessbewertungsmodell. Berlin: Beuth 2006.

Inmon 1992

Inmon, W. H.: Building the Data Warehouse. New York: Wiley and Sons 1992.

Jeckle 2000

Jeckle, M.: Konzepte der Metamodellierung - Zum Begriff Metamodell. Softwaretechnik Trends 20 (2000) 2, S. 29-30.

Jeckle et al. 2004

Jeckle, M. R. C.; Hahn, J.; Zengler, B.; Queins, S.: UML 2 glasklar. München: Hanser 2004.

Kallmeyer 1998

Kallmeyer, F.: Eine Methode zur Modellierung prinzipieller Lösungen mechatronischer Systeme. Paderborn: Bonifatius 1998. (HNI-Verlagsschriftenreihe 42).

Katz & Moon 2000

Katz, R.; Moon, Y.: Virtual Arch Type Reconfigurable Machine Tool Design: Principles and Methodology, http://erc.engin.umich.edu/publications/PubFiles/TA3/VirtualRMTReport_41.pdf, 08.10.2004.

Kemper & Eickler 1997

Kemper, A.; Eickler, A.: Datenbanksysteme - Eine Einführung. München: Oldenbourg 1997.

Kief 2005

Kief, H. B.: NC/CNC Handbuch. München: Hanser 2005.

Kindrick et al. 2000

Kindrick, J.; Hauser, M.; Barra, R.: Usage Guide for the STEP PDM Schema. Darmstadt: PROSTEP Gmbh 2000.

Kirmair & Fay 2001

Kirmair, S.; Fay, A.: Nutzung von XML zur Beschreibung von Prozessleittechnik-Hardware-Komponenten. atp - Automatisierungstechnische Praxis 43 (2001) 4, S. 55-58.

Kleiner 2003

Kleiner, S.: Föderatives Informationsmodell zur Systemintegration für die Entwicklung mechatronischer Produkte. Karlsruhe: Shaker 2003. (Forschungsberichte aus dem Fachgebiet Datenverarbeitung in der Konstruktion 15).

Klever 2001

Klever, N.: Elementarteilchen - XML-Schema: objektorientierte Dokumenttypdefinitionen. ix - Magazin für professionelle Informationstechnik 6 (2001), S. 62-66.

Koller 1985

Koller, R.: Konstruktionslehre für den Maschinenbau - Grundlagen des methodischen Konstruierens. Berlin: Springer 1985.

Kowalewski 2001

Kowalewski, S.: Modellierungsmethoden aus der Informatik. at-Automatisierungstechnik 49 (2001) 9, S. A1-A5.

Krastel 2002

Krastel, M.: Integration multidisziplinärer Simulations- und Berechnungsmodelle in PDM-Systeme. Aachen: Shaker 2002. (Forschungsberichte aus dem Fachgebiet Datenverarbeitung in der Konstruktion 11).

Krause et al. 1993

Krause, F.; Kimura, F.; Kjellberg, T.; Lu, S. C.: Product Modelling. Annals of the CIRP 42 (1993) 2, S. 695-706.

Krause et al. 1998

Krause, F.; Tang, T.; Ahle, U.: Systementwicklungen für die Integrierte Virtuelle Produktentstehung. In: Informationsverarbeitung in der Konstruktion '99 - Beschleunigung der Produktentwicklung durch EDM/PDM- und Feature-Technologie, München. Düsseldorf: VDI-Verlag 1998, S. 77-101. (VDI-Berichte 1497).

Krause et al. 2003

Krause, F.; Hayka, H.; Pasewaltdt, B.: Produktdatenbasierte Kooperation in der Produktentstehung. In: Adam, W.; Pritschow, G.; Uhlmann, E.; Weck, M. (Hrsg.): Datenmodelle in der Produktion. Düsseldorf: VDI-Verlag 2003 (VDI-Reihe 633).

Kreusch 2002

Kreusch, K.: Verifikation numerischer Steuerungen an virtuellen Werkzeugmaschinen. Karlsruhe: Shaker 2002. (Berichte aus der Steuerungs- und Regelungstechnik).

Kübler 2000

Kübler, R.: Modulare Modellierung und Simulation mechatronischer Systeme. Düsseldorf: VDI Verlag 2000. (Fortschritt-Berichte VDI 327).

Langlotz 2000

Langlotz, G.: Ein Beitrag zur Funktionsstrukturentwicklung innovativer Produkte. Karlsruhe: Shaker 2000. (Forschungsberichte RPK 2).

Lian et al. 2000

Lian, F.; Moyne, J. R.; Tilbury, D. M.: Implementation of Networked Machine Tools in Reconfigurable Manufacturing Systems. In: Symposium on Flexible Automation, Ann Arbor, Michigan, USA. 2000.

Lindemann et al. 2003

Lindemann, U.; Baumberger, C.; Freyer, B.; Gahr, A.; Ponn, J.; Pulm, U.: Entwicklung individualisierter Produkte. In: Reinhart, G.; Zäh, M. F. (Hrsg.): Marktchance Individualisierung. Berlin: Springer 2003, S. 13-29.

Lingxiang 2003

Lingxiang, X.: Wiederverwendbare Modelle zur Maschinensimulation für den Steuerungstest. München: Utz 2003. (Informationstechnik im Maschinenwesen 21).

Lobin 2003

Lobin, H.: Erweiterte Dokumentgrammatiken als Grundlage innovativer XML-Tools. IT (2003) 3, S. 143-150.

Lubell et al. 2004

Lubell, J.; Peak, R. S.; Srinivasan, V.; Waterbury, S. C.: STEP, XML and UML: Complementary Technologies. In: Proceedings of DECT 2004 (Hrsg.): ASME 2003 Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Salt Lake City, Utah, USA. 2004.

Lutz 1999

Lutz, R.: Softwaretechnik für maschinennahe Steuerungsfunktionen bei Fertigungseinrichtungen. Stuttgart: Jost-Jetter 1999. (ISW Forschung und Praxis 132).

Lyons 1995

Lyons, J.: Einführung in die moderne Linguistik. München: C. H. Beck 1995.

Mangold et al. 2003

Mangold, C.; Rantzau, R.; Mitschang, B.: Föderal: Management of Engineering Data Using a Semistructured Data Model. In: Proceedings of the 5th International Conference on Enterprise Information Systems (ICEIS), Angers, France. 2003.

Marca & McGowan 1987

Marca, D. A.; McGowan, C. L.: SADT: Structured analysis and Design Techniques. New York: McGraw-Hill 1987.

Matula 2005

Matula, M.: Netbeans Metadata Repository, http://mdr.netbeans.org/MDR-whitepaper.pdf, 11.11.2005.

Medvidovic et al. 2002

Medvidovic, N.; Rosenblum, D. S.; RedMiles, D. F.: Modeling Software Architectures in the Unified Modeling Langauge. ACM Transactions on Software Engineering and Methodology 11 (2002) 1, S. 2-57.

Meier 2001

Meier, H.: Verteilte kooperative Steuerung maschinennaher Abläufe. München: Utz 2001. (*iwb* Forschungsberichte 155).

Meier 2005

Meier, W.: Open Source Native XML Database, http://www.exist-db.org/, 12.11.2005.

Mewes und Partner 2006

Mewes und Partner: WinMOD, http://www.winmod.de, 10.10.2006.

Milberg 1992

Milberg, J.: Werkzeugmaschinen - Grundlagen. Berlin: Springer 1992.

Milberg & Ebner 1994

Milberg, J.; Ebner, K.: Verfügbarkeit von Werkzeugmaschinen. In: Abschlussbericht der AiF-Forschungsstudie 8649. Frankfurt: VDW 1994.

Montau 1996

Montau, R.: Föderatives Produktdatenmanagement anhand semantischer Informationsmodellierung. Düsseldorf: VDI Verlag 1996.

Moon & Kota 1999

Moon, Y.; Kota, S.: A Methodology for Automated Design of Reconfigurable Machines. In: Proceedings of the 32nd CIRP International Seminar on Manufacturing Systems, Leuven, Belgium. 1999, S. 297-303.

Neithardt 2004

Neithardt, W.: Methodik zur Simulation und Optimierung von Werkzeugmaschinen in der Konzept- und Entwurfsphase auf Basis der Mehrkörpersimulation. Karlsruhe: Holler 2004. (Forschungsberichte aus dem Institut für Produktionstechnik 124).

Nestrorov et al. 1997

Nestrorov, S.; Abiteboul, S.; Motwani, R.: Inferring structure in semistructured data. ACM SIGMOD Record 26 (1997) 4, S. 39-43.

Neuhaus 2003

Neuhaus, J.: Umkonfigurieren von Werkzeugmaschinen durch Plug & Play mechatronischer Module. Karlsruhe: Shaker 2003. (Berichte aus der Produktionstechnik 16).

Nikravesh 1988

Nikravesh, P. E.: Computer-Aided Analysis of Mechanical Systems. Englewood Cliffs: Prentice-Hall 1988.

Nonaka & Takeuchi 1997

Nonaka, I.; Takeuchi, H.: Die Organisation des Wissens - Wie japanische Unternehmen eine brachliegende Ressource nutzbar machen. Frankfurt: Campus 1997.

Nowacki & von Lukas 2003

Nowacki, S.; von Lukas, U.: Efficient and Convenient Federation of Product Data. In: Frick, C.; Katzenbach, A.; Krause, F.-L. (Hrsg.): Proceedings Advances in Collaborative Product Creation, Dresden. Stuttgart: Fraunhofer IRB Verlag 2003, S. 36-45.

OMG 2000

Object Management Group: Product Data Management Enablers Specification, http://www.omg.org, 10.10.2000.

OMG 2004

Object Management Group: The Object Management Group, http://www.omg.org, 12.09.2004.

OMG 2005

Object Management Group: XML Metadata Interchange (XMI) Specification, http://www.omg.org, 03.03.2005.

OMG 2006A

Object Management Group: Meta Object Facility Core Specification Version 2.0, http://www.omg.org, 06.07.2006.

OMG 2006B

Object Management Group: UML 2.0 Superstructure Specification, http://www.omg.org, 10.10.2006.

Oestereich 2005

Oestereich, B.: Analyse und Design mit UML 2 - Objektorientierte Softwareentwicklung. München: Oldenbourg 2005.

Ogden & Richards 1923

Ogden, C. K.; Richards, I. A.: The meaning of meaning - A study of the influence of language upon thought and of the science of symbolism. London: Routledge&Kegan Paul 1923.

Osmers 1998

Osmers, U.: Projektieren speicherprogrammierbarer Steuerungen mit Virtual Reality. Karlsruhe: 1998. (Forschungsberichte aus dem wbk 87).

Otter & Schweiger 2004

Otter, M.; Schweiger, C.: Modellierung mechatronischer Systeme mit Modelica. In: VDI/VDE-GMA (Hrsg.): Mechatronischer Systementwurf, Darmstadt. Düsseldorf: VDI Verlag 2004, S. 39-50. (VDI-Berichte 1842).

PDTnet 2005

PDTnet: PDTnet, http://www.pdtnet.org, 10.09.2005.

Pahl et al. 2004

Pahl, G.; Beitz, W.; Feldhusen, J.; Grote, K. H.: Konstruktionslehre. Berlin: Springer 2004.

Pan & Smith 2003

Pan, C.; Smith, S.: Extracting Geometrical Data from CAD STEP Files. In: Proceedings of ASME 2003 - Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Chicago, IIlinois, USA. 2003.

Papakonstantinou et al. 1995

Papakonstantinou, Y.; Garcia-Molina H.; Widom, J.: Object Exchange across Heterogeneous Information Sources. In: YU, P. S.; Chen, A. L. P. (Hrsg.): 11th Conference on Data Engineering, Taipeh, Taiwan. Washington: IEEE Computer Society 1995, S. 251-260.

Paul 1995

Paul, G.: Produktdatenverwaltungssysteme - Methoden und Werkzeuge zur Unternehmensintegration. CIM Management 11 (1995) 4.

Probst et al. 2003

Probst, G.; Raub, S.; Rombardt, K.: Wissen managen. Wie Unternehmen ihre wertvollste Ressource optimal nutzen. Wiesbaden: Gabler 2003.

Pühl 1999

Pühl, S.: Engineering Object Management System. Karlsruhe: Shaker 1999. (wzl Berichte aus der Produktionstechnik 24).

Rechenberg 2003

Rechenberg, P.: Zum Informationsbegriff der Informationstheorie. Informatik Spektrum 26 (2003) 5, S. 317-326.

Redman 1992

Redman, T.: Data Quality - Management and Technology. New York: Bantham Books 1992.

Reichwald & Piller 2002

Reichwald, R.; Piller, F. T.: Der Kunde als Wertschöpfungspartner - Formen und Prinzipien. In: Albach, H.; Kaluza, B.; Kersten, W. (Hrsg.): Wertschöpfungsmanagement als Kernkompetenz. Wiesbaden: Gabler 2002.

Reinhart et al. 1998

Reinhart, G.; Schneider, C.; Weißenberger, M.; Sprenzel, A.; Meinlschmidt, J.: Kooperative Entwicklung technologisch komplexer Produkte. Konstruktion 50 (1998) 9, S. 15-22.

Reinhart et al. 1999

Reinhart, G.; Bauer, L.; Meier, H.; Wagner, P.; Weißenberger, M.: Vernetzte Entwicklung komplexer mechatronischer Produkte. ZWF 94 (1999) 4, S. 191-194.

Reinhart et al. 2001A

Reinhart, G.; Anton, O.; Lercher, B.: Babylon der Fachsprachen. IT - Industrielle Informationstechnik (2001) 11/12, S. 48-49.

Reinhart et al. 2001B

Reinhart, G.; Anton, O.; Lercher, B.: Funktionsorientiertes Sichtenmodell für die Entwicklung mechatronischer Systeme. VDI-Z 143 (2001) 11/12, S. 67-70.

Reinhart et al. 2001C

Reinhart, G.; Oertli, T.; Zeller, W.: Auslegung von NC-Antrieben unter Berücksichtigung strukturdynamischer und prozessbedingter Wechselwirkungen. In: Schwingungen in Antrieben 2001, Würzburg. Düsseldorf: VDI Verlag 2001, S. 187-207. (VDI-Berichte 1630).

Reinhart et al. 2002A

Reinhart, G.; Anton, O.; Lercher, B.; Oertli, T.: Funktionsmodellierung und Dynamikoptimierung beim mechatronischen Systementwurf. In: Reinhart, G. (Hrsg.): Mechatronische Produktionssysteme - Genauigkeit gezielt entwickeln, Garching. München: Herbert Utz 2002, S. 1-17. (*iwb* Seminarberichte 60).

Reinhart et al. 2002B

Reinhart, G.; Ehrenstrasser, M.; Pörnbacher, C.; Wünsch, G.: Virtuelle Werkzeugmaschinen für die Simulation. wt-online 92 (2002) 5, S. 205-209.

Reinhart et al. 2002C

Reinhart, G.; Anton, O.; Lercher, B.: Sichtenorientierte Störungsmodellierung an Werkzeugmaschinen. Industriemanager 18 (2002) 1, S. 35-39.

Reinhart & Cisek 2003

Reinhart, G.; Cisek, R.: Mit Mobilität zur wandlungsfähigen Produktion. In: Reinhart, G.; Zäh, M. F. (Hrsg.): Marktchance Individualisierung. Berlin: Springer 2003, S. 75-88.

Richert et al. 2003

Richert, F.; Rückert, J.; Schloßer, A.: Vergleich von Modelica und Matlab anhand der Modellbildung eines Dieselmotors. at-Automatisierungstechnik 51 (2003) 6, S. 247-254.

Roller & Schäfer 2002

Roller, D.; Schäfer, D.: Elektrotechnik CAD - CAE Systeme der dritten Generation. In: Roller, D.; Schäfer, D. (Hrsg.): Berichte aus der Konstruktionstechnik. Stuttgart: Shaker 2002.

Roth 2001

Roth, K.: Konstruieren mit Konstruktionskatalogen. Berlin: Springer 2001.

Rumpe 1996

Rumpe, B.: Formale Methodik des Entwurfs verteilter objektorientierter Systeme. München: Utz 1996.

Rumpe & Sandner 2001

Rumpe, B.; Sandner, R.: UML - Unified Modeling Language im Einsatz Teil 1. at - Automatisierungstechnik 49 (2001) 9, S. A6-A14.

Sabbah 2000

Sabbah, K.: Methodische Entwicklung störungstoleranter Steuerungen. München: Herbert Utz 2000. (*iwb* Forschungsberichte 139).

Schelberg 1994

Schelberg, H. J.: Objektorientierte Projektierung von SPS-Software. Karlsruhe: Schnelldruck Ernst Grässer 1994. (wbk Forschungsberichte 57).

Schernikau 2001

Schernikau, J.: Gestaltung von mechatronikgerechten Organisationen in der Produktentwicklung. Karlsruhe: Shaker 2001. (wzl Berichte aus der Produktionstechnik 8).

Schielen 1986

Schiehlen, W.: Technische Dynamik. Stuttgart: Teubner 1986.

Schimkat 2003

Schimkat, R. D.: Techniken und Aspekte zur Realisierung proaktiver Informationssysteme. Tönning: Der andere Verlag 2003.

Schneider 2000

Schneider, C.: Strukturmechanische Berechnungen in der Werkzeugmaschinenkonstruktion. München: Utz 2000. (*iwb* Forschungsberichte 144).

Schöttner 1999

Schöttner, J.: Produktdatenmanagement in der Fertigungsindustrie. Wien: Carl Hander 1999.

Selic et al. 1994

Selic, B.; Gullekson, G.; Ward, P. T.: Real-Time Object-Orientied Modeling. New York: Wiley 1994.

Sellentin 1999

Sellentin, J.: Konzepte und Techniken der Datenversorgung für komponentenbasierte Informationssysteme. Universität Stuttgart, Fakultät Informatik: Dissertation 2000.

Shiling 2001

Shiling, M.: Methodische Entwicklung und rollenbasierte Integration von Komponentenframeworks. München: Dissertation Technische Universität München 2001.

Sielaff 2003

Sielaff, F.: Integrating Software Design in the Mechanical Product Development Process. In: Frick, C.; Katzenbach, A.; Krause, F. L. (Hrsg.): Proceedings of Advances in Collaborative Product Creation, Dresden. Stuttgart: Fraunhofer IRB Verlag 2003, S. 142-150.

Spur & Krause 1997

Spur, G.; Krause, F. L.: Das virtuelle Produkt: Management der CAD-Technik. München: Hanser 1997.

Staab 2002

Staab, S.: Wissensmanagement mit Ontologien und Metadaten. Informatik Spektrum 25 (2002) 3, S. 194-209.

Steger 2001

Steger, A.: Diskrete Strukturen. Berlin: Springer 2001.

Suciu 1997

Suciu, D.: Management of Semistructured Data. ACM SIGMOD Record 26 (1997) 4.

Suciu 1998

Suciu, D.: An Overview of Semistructured Data. SIGACT News 29 (1998) 4.

Svensson & Crnkovic 2002

Svensson, D.; Crnkovic, I.: Information Management for Multi-Technology Products. In: International Design Conference - Design 2002. Dubrovnik: 2002.

Svensson 2003

Svensson D.: Towards Product Structure Management in Heterogeneous Environments. Göteborg: Chalmers 2003.

Technomatix 2005

Technomatix 2005: eM-Workplace, http://www.technomatix.de, 11.11.2005.

Tilly & Tilkov 2004

Tilly, M.; Tilkov, S.: Schemasprache Relax NG - Entspannung pur. ix - Magazin für professionelle Informationstechnik 10 (2004), S. 124-127.

Tomaszunas 1998

Tomaszunas, J. J.: Komponentenbasierte Maschinenmodellierung zur Echtzeit-Simulation für den Steuerungstest. München: Utz 1998. (Informationstechnik im Maschinenwesen 3).

Trippner 2002

Trippner, D.: Vorgehensmodell zum Management von Produktdaten in komplexen und dynamischen Produktentwicklungsprozessen. Karlsruhe: Shaker 2002. (Forschungsberichte aus dem RPK 5).

Tzitzikas et al. 2001

Tzitzikas, Y.; Spyratos, N.; Constantopoulos, P.: Mediators over Ontology-based Information Sources. In: Second International Conference on Web Information Systems Engineering (WISE 2001), Kyoto. 2001.

Tzitzikas 2002

Tzitzikas, Y.: Colaborative Ontology-based Information Indexing and retrieval. Heraklion: University of Crete, Department of Computer Science, 2002.

Tzitzikas et al. 2002

Tzitzikas, Y.; Spyratos, N.; Constanopoulos, P.: Query Translation for Mediators over Ontoloy-based Information Sources. In: Proceedings of the Second Hellenic Conference on AI: Methods and Applications of Artificial Intelligence, Thessaloniki, Greece . London: Springer 2002, S. 423-436. (Lecture Notes in Computer Science 2308).

Tzitzikas et al. 2004

Tzitzikas, Y.; Meghini, C.; Spyratos, N.: A Unifying Framework for Flexible Information Access in Taxonomy-based Sources. In: Christiansen, H.; Hacid, M. S.; Andreasen, T. (Hrsg.): Flexible Query Answering Systems: 6th International Conference, FQAS 2004, Lyon, France. Berlin: Springer 2004, S. 161-174. (Lecture Notes in Computer Science 3055).

Tzitzikas et al. 2005

Tzitzikas, Y.; Spyratos, N.; Constantopoulos, P.: Mediators over Taxonomy-based Information Sources. VLDB Journal 14 (2005) 1, S. 112-136.

Uhlig 2002

Uhlig, V.: Informationstechnische Unterstützung von Entwicklungskooperationen. Garbsen: PZH Produktionstechnisches Zentrum GmbH 2002. (Berichte aus dem IFW 3).

Ungerer & Fischer 2002

Ungerer, M.; Fischer, M.: Zulieferanbindung in der Kabelbaumentwicklung durch STEP AP 212. In: Roller, D.; Schäfer, D. (Hrsg.): Berichte aus der Konstruktionstechnik. Stuttgart: Shaker 2002, S. 91-97.

Unland 1995

Unland, R.: Objektorientierte Datenbanken - Konzepte und Modelle. Bonn: Thomson Publishing 1995.

VDI 2221 1993

VDI-Richtlinie 2221: Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte. Berlin: Beuth 1993.

VDI/VDE 3681 2005

VDI/VDE 3681: Einordnung und Bewertung von Beschreibungsmitteln aus der Automatisierungstechnik. Berlin: Beuth 2005.

Varnhagen 2000

Varnhagen, V.: Qualitätsplanungs- und Qualitätscontrolling-Systematik im Rahmen des Simultaneous Engineering. Karlsruhe: Shaker 2000. (Berichte aus der Produktionstechnik 29).

VDW 2003

Verein Deutscher Werkzeugmaschinenfabriken e.V. (Hrsg.): Simulationstechniken zur Verkürzung der Durchlaufzeiten bei der Entwicklung von Werkzeugmaschinen. Frankfurt: 2003.

Vlist 2003

van der Vlist, E.: Relax NG. Beijing: O'Reily 2003.

Vogel 2000

Vogel, U.: Informationsmodell für die Organisation des Produktentwicklungsprozesses im virtuellen Unternehmen. Karlsruhe: Shaker 2000.

W3C 2001

World Wide Web Consortium: Semantic Web, http://www.w3.org/2001/sw/, 23.06.2005.

W3C 2004A

World Wide Web Consortium: Extensible Markup Language (XML) 1.0, http://www.w3.org/TR/2004/ REC-xml-20040204, 12.01.2006

W3C 2004B

World Wide Web Consortium: OWL Web Ontology Language, http://www.w3.org/TR/owl-ref/, 12.01.2006

W3C 2005

World Wide Web Consortium: Resource Description Framework (RDF), http://www.w3.org/ RDF/, 23.07.2005.

Wagner 1995

Wagner, P.: Kostenanalysen und resultierende Neuerungsansätze. In: Reinhart, G. (Hrsg.): Installationstechnik an Werkzeugmaschinen - Analysen und Konzepte, Garching. München: Utz 1995, S. 13-36.

Wagner 1997

Wagner, M.: Steuerungsintegrierte Fehlerbehandlung für maschinennahe Abläufe. Berlin: Springer 1997. (*iwb* Forschungsberichte 106).

Weber 2004

Weber, V.: Dynamisches Kostenmanagement in kompetenzzentrierten Unternehmensnetzwerken. München: Utz 2004. (*iwb* Forschungsberichte 183).

Weck & Possel-Dölken 2003

Weck, M.; Possel-Dölken, F.: Vertikale Integration - vom ERP System zur Werkzeugmaschine und zurück. wt-online 93 (2003) 5, S. 366-373.

Weck & Teipel 1977

Weck, M.; Teipel, K.: Dynamisches Verhalten spanender Werkzeugmaschinen. Berlin: Springer 1977.

Weissmahr 1991

Weissmahr, B.: Ontologie. Stuttgart: Kohlhammer 1991.

Wiederhold 1992

Wiederhold, G.: Mediators in the architecture of future information systems. IEEE Computers 25 (1992) 3, S. 38-49.

Wollschlaeger 1999

Wollschlaeger, M.: XML-Beschreibungen als Basis für WWW-basierte Managementlösungen von Feldbuskomponenten. In: VDI/VDE-Gesellschaft Mexx- und Automatisierungstechnik (Hrsg.): GAM-Fachtagung Industrielle Automation und Internet/Intranet-Technologie, Langen. Düsseldorf: VDI Verlag 1999, S. 57-66. (VDI-Berichte 1515).

WZM-20xx 2005

Forschungszentrum Karlsruhe (PTKA): Werkzeugmaschine 20XX - Szenariengestützte Instrumente zur Strategieentwicklung,Transferbündelung und begleitende Wirkungsanalyse, http://www.wzm-initiative.de, 12.03.2005.

Zaeh & Poernbacher 2005

Zaeh, M. F.; Poernbacher, C.: A Model-Based Method to Develop PLC Software for Machine Tools. In: Annals of the CIRP 54/1. 2005, S. 371-374.

Zäh et al. 2002

Zäh, M.; Englberger, G.; Oertli, T.: Mechatronic System Simulation of NC Drives in Machine Tool Design. DriveWorld - The Drive Technology Magazine of the SEW-EURODRIVE Group (2002) 2, S. 18-21.

Zäh et al. 2003

Zäh, M. F.; Wünsch, G.; Pörnbacher, C.; Ehrenstrasser, M.: Emerging Virtual Machine Tools. In: ASME (Hrsg.): Proceedings of DECT 2003 - 29th Design Automation Conference, Chicago, Illinois, USA. 2003.

Zäh et al. 2004

Zäh, M. F.; Englberger, G.; Oertli, T.; Siedl, D.: Strukturdynamik und Mechatronik in der Werkzeugmaschinenentwicklung. In: Zäh, M. F. (Hrsg.): Strukturdynamik von Werkzeugmaschinen, Garching. München: Utz 2004, S. 2.1-2.26. (*iwb* Seminarberichte 70).

Zaeh & Lercher 2006

Zaeh, F. M.; Lercher, B.: Towards Data Integration in Virtual Machine Tool. In: German Academic Society for Production Engineering (Hrsg.): Annals of the German Academic Society for Production Engineering. Hannover: WGP e.V. 13 (2006) 1, S. 207-209.

Zeigler & Praehofer 2000

Zeigler, B. P.; Praehofer H.; Tag Gon, K.: Theory of Modeling and Simulation. San Diego: Academic Press 2000.

XML:DB Initiative 2005

Initiative for XML Databases: Initiative for XML Databases, http://xmldb-org.sourceforge.net/,10.09.2005.

11 Anhang

11.1 Begriffsdefinitionen

Aufgrund des zum Teil formalen Charakters einiger Abschnitte ist ein exaktes Verständnis der verwendeten Begriffe erforderlich. Im Folgenden werden aus diesem Grund Definitionen einzelner Begriffe aufgeführt, die für das Verständnis dieser Arbeit notwendig sind.

System

Nach [DIN 19226 1984] ist ein *System* eine abgegrenzte Anordnung miteinander in Verbindung stehender Gebilde, die Gegenstände oder Denkmethoden und deren Ergebnisse umfassen können. Diese Anordnung wird durch eine Hüllfläche (Systemgrenze) von ihrer Umgebung abgegrenzt. Durch die Hüllfläche werden Verbindungen des Systems mit seiner Umgebung geschnitten. Die Verbindungen sind so zu wählen, dass die damit übertragenen Größen und deren Beziehungen zueinander das dem System eigentümliche Verhalten beschreiben.

In [ZEIGLER & PRAEHOFER 2000] wird die Unterscheidung zwischen der Struktur und dem Verhalten eines Systems hervorgehoben. Mit der Struktur ist der innere Aufbau des Systems gemeint. In der Systemtheorie existieren zwei Konzepte, um Systemstrukturen aufbauen zu können. Die Dekomposition befasst sich mit der Fragestellung, wie ein System in kleinere Komponenten untergliedert werden kann. Die Komposition hingegen geht der Fragestellung nach, wie aus einer Menge einzelner Komponenten ein System aufgebaut werden kann. Das Verhalten eines Systems wird durch die Beziehung zwischen den Ein- und Ausgangsgrößen beschrieben.

Entität

Eine *Entität* ist ein individuelles und identifizierbares Exemplar von Dingen, Personen oder Begriffen der realen oder der Vorstellungswelt und wird durch Eigenschaften beschrieben. Für Entität wird synonym oft der Begriff Objekt verwendet [BALZERT 1996].

Tupel

Der Begriff *n-Tupel* stammt aus der Mathematik und bezeichnet eine geordnete Sequenz von Elementen. *n* stellt hierbei die Anzahl der Elemente des Tupels dar, im Falle von n = 2 spricht man auch von einem geordneten Paar oder einfach von einem Tupel. Als Schreibweise wird $(e_1, e_2, ..., e_n)$ gewählt.

Modell

Nach [KOWALEWSKI 2001] ist der Maschinenbau allgemein vom Modellierungsbegriff der Naturwissenschaften geprägt, nämlich der Beschreibung von physikalischen oder chemischen Phänomenen. Zweck der Modellbildung ist es dabei, genügend Verständnis über die Zusammenhänge zwischen den Einflussfaktoren und den problemabhängigen Zielgrößen zu gewinnen, um die beschriebenen Systeme gezielt beeinflussen zu können. Die Informatik verbindet mit dem Begriff Modellierung dagegen die Beschreibung von Anforderungen an ein zu ent-

werfendes Rechensystem oder von Entwürfen, die diese Anforderungen erfüllen sollen. Der Modellierungszweck besteht bei der Analyse darin, zu einem besseren Verständnis der gewünschten Eigenschaften und der geforderten Funktionalität zu kommen. Die Entwurfsmodelle dienen dazu, Lösungen zur Erreichung dieser Eigenschaften zu untersuchen, weiterzuentwickeln und anderen Personen zu vermitteln. Die Komplexität rührt also nicht wie im allgemeinen Maschinenbau von Naturphänomenen her, sondern liegt in der menschlichen Schwierigkeit begründet, komplexe Aufgabenstellungen bzw. Funktionalitäten nachvollziehbar, vollständig und widerspruchsfrei zu formulieren.

Daten und Datensatz

Die Informatik definiert *Daten* als maschinenlesbare, bearbeitbare Repräsentationen von Informationen. Daten werden in Zeichen kodiert, wobei deren Aufbau strengen Regeln unterliegt. Mit Datensatz bezeichnet man mehrere Daten, die aufgrund ihrer inhaltlichen Gemeinsamkeiten zu einer Einheit zusammengefasst werden.

Datenqualität

Die *Datenqualität* bezeichnet die Güte von Informationen. Im Wesentlichen sind damit Qualitätskriterien wie Genauigkeit, Vollständigkeit, Korrektheit und Widerspruchsfreiheit gemeint [REDMAN 1992]. Die Frage nach der Qualität von Daten bezieht sich auf die Aussage, wie verlässlich eine Information ist und inwieweit sie als Grundlage für Entscheidungen verwendet werden kann. Die Ursachen für eine schlechte Datenqualität liegen oft in Designfehlern wie homonymen und synonymen Bezeichnungen, unvollständigen oder falschen Daten und unzureichenden Kontrollmechanismen bei der Eingabe.

Dokument

Ein *Dokument* ist eine bei der Entwicklung eines Systems verwendete, abgegrenzte Einheit zur Beschreibung eines Aspekts oder eines Ausschnitts des zu erstellenden Systems. Ein Dokument besitzt eine exakt definierte, abstrakte Syntax und eine konkrete Darstellungsform [RUMPE 1996]. Daten, Modelle und Informationen werden mit Dokumenten transportiert.

Datenbasis und Datenbank

Unter dem Begriff der *Datenbasis* wird eine bestimmte, abgrenzbare Menge von Daten verstanden. Diese sind derart organisiert, dass ein systematischer Zugriff erfolgen kann. In der Literatur wird mitunter zwischen Datenbasis und Datenbank unterschieden. In diesem Fall stellt die Datenbank eine konkrete, implementierte systematische Organisation der Daten auf einem Rechner dar, die anhand unterschiedlicher Datenmodelle beschrieben werden. Der Begriff *Datenbasis* ist somit eine Verallgemeinerung des Begriffs *Datenbank* ohne Angabe über die zu Grunde liegende Datenstrukturierung bzw. -organisation. Ein Datenbanksystem oder Datenbankmanagementsystem besteht aus der Datenbank selbst und den Funktionen für deren Verwaltung und Organisation.

Relationales und objekt-orientiertes Datenmodell

Jedes Datenbanksystem basiert auf einem *Datenmodell*, in welchem festgelegt wird, welche Eigenschaften die Datenelemente besitzen, welche Struktur die Datenelemente haben dürfen, welche Konsistenzbedingungen einzuhalten sind und welche Operationen zum Speichern, Suchen, Ändern und Löschen von Datenelementen existieren.

Das relationale Datenmodell wurde erstmals in [CODD 1970] veröffentlicht und basiert auf dem mathematischen Konzept der Relation. Sind $A_1, A_2, ..., A_n$ endliche Mengen, so heißt die Menge aller Kombinationen ihrer Elemente ihr kartesisches Produkt $[A_1 \times A_2 \times ... \times A_n]$. Die Elemente von kartesischen Produkten heißen Tupel. Jede Relation $R \subseteq A_1 \times A_2 \times ... \times A_n$ kann als Tabelle mit dem Namen *R* dargestellt werden. Die Spalten tragen die Namen der Attribute A_1 bis A_n, in den Zeilen sind die Tupel aufgeführt. Da Relationen Mengen sind, ist das mehrfache Vorkommen eines Tupels ausgeschlossen [BALZERT 1999].

Im relationalen Modell ist die einzige Möglichkeit der Darstellung von Entitäten das Tupel. Ein Tupel besteht aus einer festen Anzahl von atomaren, nicht veränderlichen Werten, wie beispielsweise der Zahl *2*. Atomar bedeutet, dass die Werte nicht aus komplexeren Strukturen, beispielsweise aus weiteren Werten, zusammengesetzt sind. Das objektorientierte Datenmodell erlaubt demgegenüber eine sehr viel flexiblere Strukturbeschreibung [KEMPER & EICKLER 1997]. Reale Gegenstände werden hier direkt durch Datenbankobjekte repräsentiert. Ihre Identifikation erfolgt über eindeutige und unveränderliche Objektidentifikatoren, welche vom Datenbanksystem vergeben werden. Ein Datenbankobjekt kann im Gegensatz zum Tupel weitere Bestandteile haben, die selbst wieder Objekte sind [UNLAND 1995] und diese mittels Vererbung an andere Objekte weitergeben.

Kartesisches Produkt

Das *kartesische Produkt* zweier Mengen *A* und *B* ist die Menge aller geordneten Paare *(a, b)*, wobei *a* Element aus *A* und *b* Element aus *B* ist. Formal lässt sich dies wie folgt beschreiben: $A \times B := \{(a, b); a \in A \text{ und } b \in B\}$.

Vereinungsmenge

Mit dem Symbol *U* ist eine nichtleere Menge von Mengen bezeichnet. Dann ist die *Vereinigungsmenge* die Menge der Elemente, die in mindestens einem Element aus *U* enthalten ist. Formal lässt sich dies wie folgt beschreiben: $\bigcup U := \{x; \exists a \in U : x \in a\}$.

Graph

Ein *Graph G* ist ein Tupel (V, E), wobei V eine (endliche) nichtleere Menge von Knoten (engl. vertices) ist. Die Menge E ist eine Teilmenge der zweielementigen Teilmengen von *V*, also $E \subseteq \binom{V}{2} := \{\{x, y\} \mid x, y \in V, x \neq y\}$. Die Elemente der Menge E bezeichnet man als Kanten (engl. Edges). Ein Graph wird dargestellt, indem man jeden Knoten des Graphen durch einen Punkt repräsentiert und die Punkte genau dann durch einen Strich verbunden werden, wenn im Graphen die entsprechenden Knoten durch eine Kante verbunden sind [Steger 2001].

Formale Sprachen

Die Grundelemente *formaler Sprachen* sind einfache, natürlich sprachliche Symbole wie z. B. Buchstaben und Ziffern. Eine Zeichenkette (oder auch ein Wort) ist eine endliche Folge von Symbolen, die ohne Zwischenraum hintereinander geschrieben werden. So sind z. B. *a*, *b* und *c* Symbole und *abcd* ist eine Zeichenkette. Ein Alphabet ist eine endliche Menge von Symbolen. Eine formale Sprache ist eine Menge von Zeichenketten, die aus den Symbolen eines beliebigen Alphabets zusammengesetzt sind. Die Zeichenketten sind nach bestimmten Regeln aufgebaut, welche die Kombination der Symbole aus dem Alphabet genau definieren. Den hierdurch beschriebenen formalen Aufbau der Zeichenketten bezeichnet man als Syntax der Sprache. Die Syntax einer formalen Sprache wird durch eine Grammatik definiert [HOPCROFT & ULLMANN 1994].

Semantik

Als *Semantik* wird die Bedeutung sprachlicher Ausdrücke beziehungsweise von Zeichen im Allgemeinen verstanden. Die Aufgabe der formalen Semantik besteht darin, mithilfe logisch-mathematischer Methoden Regeln zu formulieren, durch die Ausdrücke und Sätze künstlicher (logischer) Sprachen gedeutet werden können [BROCKHAUS 2006].

Application Programming Interface (API)

Eine *API* ist eine Programmierschnittstelle, die ein Softwaresystem anderen Softwaresystemen zur Verfügung stellt, um auf interne Daten zugreifen zu können. Die API definiert zu diesem Zweck eine Menge von Operationen und stellt diese auf Quelltextebene einer spezifischen Programmiersprache zur Verfügung.

Applikationsserver

Ein *Applikationsserver* ist eine Laufzeitumgebung für verteilte transaktionale, hochskalierbare Geschäftsanwendungen. Er stellt den Anwendungen Dienstleistungen zur Verfügung, damit diese auf Instanzen verschiedener Datenbanken zugreifen können. Es ist die Aufgabe eines Applikationsservers, die Applikationslogik des zu entwickelnden Informationssystems zu verwalten [DENNINGER & PETERS 2000]. Welche Dienste ein Applikationsserver unterstützen sollte, ist in der Literatur nicht genau festgelegt. Es ist charakteristisch für Applikationsserver, dass die bereitgestellten Dienste nicht direkt vom Anwender genutzt werden können. Vielmehr bieten sie erst die Grundlage für anwendungsspezifische Softwaresysteme, die letztendlich die gewünschte Funktionalität realisieren.

Information

Information ist ein sehr weitläufiger Begriff und damit schwer abzugrenzen. Aus Sicht der Informatik ist Information eine gedeutete Nachricht bzw. Mitteilung. Dabei gilt es, prinzipiell zwischen syntaktischer und semantischer Information zu unterscheiden. Die syntaktische Information lässt sich als ein Maß für die kürzeste Kodierung einer Nachricht definieren. Sie bezeichnet die reine Informationsmenge, ohne Rücksicht auf die Bedeutung der Information. Sie ist bestimmt durch ein Alphabet und durch die Auftrittswahrscheinlichkeit seiner Zeichen. Bei der semantischen Information steht hingegen die Bedeutung einer Nachricht für den Emp-

fänger im Vordergrund. Entscheidend ist, dass sie einen Empfänger braucht, da erst beim Empfänger die Bedeutung entsteht [RECHENBERG 2003].

Wissen

Wissen ist die Gesamtheit der Kenntnisse und Fähigkeiten, die Individuen zur Lösung von Problemen einsetzen. Dies umfasst sowohl theoretische Erkenntnisse als auch praktische Alltagsregeln und Handlungsanweisungen. Wissen stützt sich auf Daten und Informationen, ist jedoch im Gegensatz zu diesen an Personen und an einen Kontext gebunden. [PROBST ET AL. 2003]. Ein wesentlicher Aspekt des Wissens wird in [NONAKA & TAKEUCHI 1997] aufgegriffen, nämlich die Unterscheidung von explizitem sowie implizitem Wissen. Implizites Wissen lässt sich nur bedingt formalisieren und ist höchst subjektiv. Explizites Wissen ist hingegen beschreibbares, formalisierbares Wissen, das standardisiert, strukturiert und methodisch in sprachlicher Form in Dokumenten, Datenbanken oder Patenten angelegt werden kann.

Produktstruktur und Sichten

Der Begriff *Produktstruktur* wird in [DIN 199 1977] als ein produktdarstellendes Modell definiert, das die Gesamtheit der nach bestimmten Gesichtspunkten (z. B. Fertigung, Montage, Funktion) festgelegten Beziehungen zwischen Baugruppen und Einzelteilen eines Produktes beschreibt. Die Beziehungen werden durch Relationen der Art *gehört zu* oder *besteht aus* gebildet. Produktstrukturen werden fachbereichs- und aufgabenspezifisch aufgebaut. Zu einem Produkt kann es demnach verschiedene Produktstrukturen geben. In diesem Fall spricht man von Sichten [EIGNER & STELZER 2001].

Repository

Unter einem *Repository* wird allgemein eine Datenbank verstanden, die Informationen über Entwicklungsdokumente enthält, welche von bestimmten Softwarewerkzeugen erstellt wurden [BERNSTEIN & DAYAL 1994]. Bei den Dokumenten kann es sich um Source-Code, Softwaremodelle oder einfache Textdokumente handeln. Die Aufgabe des Repositories ist es, die Struktur und den Inhalt der Entwicklungsdokumente zu verwalten. So ermöglicht ein Repository für den Anwendungsbereich der Softwareentwicklung die Verwaltung von Datenbankschemata, Schnittstellendefinitionen, Source-Code, Hilfetexten und Textdokumenten. Die verschiedenen Entitäten eines Anwendungsbereichs werden in einem Informationsmodell abgebildet und miteinander in Beziehung gesetzt. Das Repository stellt Funktionen bereit, um die Entitäten des Informationsmodells zu beschreiben, zu verwalten und für den Anwender abrufbar zu machen. Charakteristisch für Repositorysysteme ist es, dass der Kontext der Daten, also deren Semantik, im System verwaltet wird. Aus der Verbindung von Datenobjekt und dessen Semantik ergibt sich die Information zur Definition von Sichten, welche die Aufgabengebiete der einzelnen Fachbereiche beschreiben. Repositories sind demnach durch ihr Informationsmodell eng an einen bestimmten Fachbereich gebunden und dienen als zentrale Datenbasis für die Integration mehrerer Softwarewerkzeuge [BERNSTEIN 1998].

Produktmodell

Ein *Produktmodell* besteht aus einer Produktstruktur sowie Dokumenten und Dokumentenstrukturen. Produktmodelle haben die Zielsetzung, Produkte mit ihren für den gesamten Lebenszyklus relevanten Informationen digital abzubilden [PAUL 1995]. Sie stellen eine gemeinsame, abstrakte Datenbasis für alle produktbeschreibenden Informationen, die so genannten Produktdaten, dar. Das Ziel der Produktdatenmodellierung besteht in der Bereitstellung eines Informationsbestandes für Produktdaten, um damit die unterschiedlichen Tätigkeiten im Produktentwicklungsprozess zu unterstützen. Integrierte Informationsmodelle enthalten Produktdaten über den gesamten Produktlebenszyklus [KRAUSE ET AL. 1993].

11.2 Modellierungssprachen für die Softwareentwicklung

11.2.1 Realtime Object Oriented Modeling (ROOM)

Unter *Realtime Object Oriented Modeling* (ROOM) wird eine objektorientierte Modellierungsmethode verstanden, die Ende der achtziger Jahre von Bell-Northern Research Ltd. in Kanada entworfen wurde. Das Ziel war es, eine Architektursprache zu realisieren, die den gesamten Entwicklungsprozess im Telekommunikationsbereich abdeckt. Insbesondere galt es, den Übergang von der Konzept- zur Kodierungsphase zu schließen und somit möglichst frühzeitig ausführbare Modelle für den Funktions- und Systemtest einsetzen zu können. 1990 wurde mit dem Softwarewerkzeug *ObjecTime* die ROOM-Methode erstmals einem breiten Anwenderkreis zur Verfügung gestellt.

ROOM [SELIC ET AL. 1994] eignet sich besonders für die Modellierung von eingebetteten Systemen. Die Methode unterscheidet dabei die zwei Modelldimensionen *Struktur* und *Verhalten*. In der Struktur-Sicht wird die Topologie des zu entwickelnden Systems dargestellt. Dabei wird das System aus einer Black-Box-Sicht betrachtet, in der die Kommunikation verschiedenener Komponenten beschrieben wird. Die Komponenten werden als *Aktoren* bezeichnet und stellen gekapselte, voneinander unabhängige Subsysteme dar (siehe Abbildung 11-1).

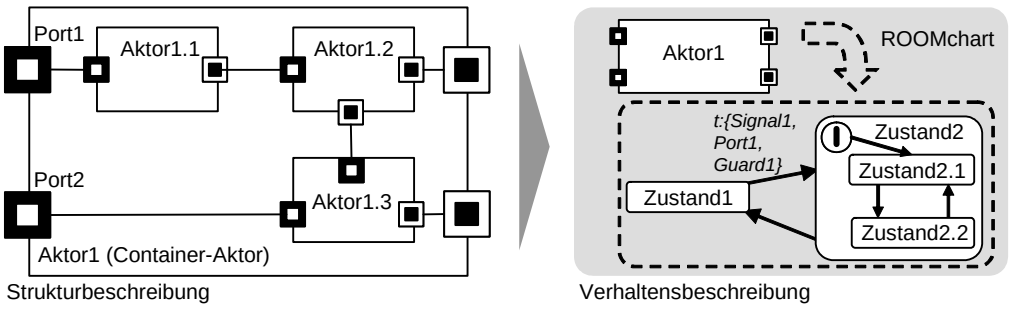


Abbildung 11-1: Systemmodellierung in ROOM

Ein Aktor wird als Container-Aktor bezeichnet, wenn er weitere Aktoren enthält.

In Abbildung 11-1 dient *Aktor1* als Container für die Aktoren *Aktor1.1*, *Aktor1.2* und *Aktor1.3*. Die Kommunikation der Aktoren erfolgt asynchron oder synchron anhand von Nach-

richten. Diese bestehen aus einem Signal sowie einem optionalen Datum und einer Prioritätsangabe. Die Nachrichten werden in Protokollen definiert. Der Nachrichtenaustausch erfolgt über die Schnittstellen der Aktoren, die so genannten *Ports*.

Das Verhalten der Aktoren wird mit *ROOMcharts* beschrieben. Dies sind erweiterte, endliche Zustandsgraphen [BALZERT 1996]. Sie basieren auf den Statecharts von [HAREL 1987]. Hier unterscheidet man prinzipiell zwei Ebenen. Die Abstraktionsebene umfasst Zustände, Transitionen und Transitionsbedingungen. In der Detailebene werden Funktionen in C++ oder in der ROOM-eigenen Sprache *ROOM Programming Language* (RPL) definiert. ROOMcharts ermöglichen hierarchische Zustände, wobei die parallelen Zustände der Statecharts nicht übernommen wurden. ROOM-Modelle befinden sich immer in genau einem aktiven Zustand. Zustandsübergänge werden durch Transitionsbedingungen definiert. Sie werden mit *t:{….}* angegeben und spezifizieren Signale und Bedingungen, die einen Zustandsübergang auslösen. Ein Beispiel einer Funktion der Detailebene ist die in Abbildung 11-1 dargestellte *Guard-Funktion*. Trifft das *Signal1* über den *Port1* ein und liefert die *Guard-Funktion* den Wert *True* zurück, wird der Zustandsübergang ausgelöst. Mit einem Zustandsübergang kann optional auch ein Befehlsblock verknüpft sein. Dieser ist ebenfalls in C++ oder RPL definiert und wird dann abgearbeitet, wenn ein Zustandsübergang ausgelöst wurde. Eine ausführliche Beschreibung der einzelnen Modellelemente ist in [SELIC ET AL. 1994] zu finden.

11.2.2 Unified Modeling Language (UML)

Die UML ist eine graphische Sprache und Notation zur Spezifikation, Visualisierung, Konstruktion und Dokumentation von Modellen für Softwaresysteme, Geschäftsprozesse sowie technische Systeme [OESTEREICH 2005]. Sie stellt mehrere, einander ergänzende Diagrammtypen zur Darstellung verschiedener Sichten zur Verfügung und liegt mittlerweile in der Version 2.0 vor. Die UML wird von der *Object Management Group* (OMG) [OMG 2004] standardisiert und weiterentwickelt. In Abbildung 11-2 sind einige der wesentlichen Diagrammtypen zu sehen.

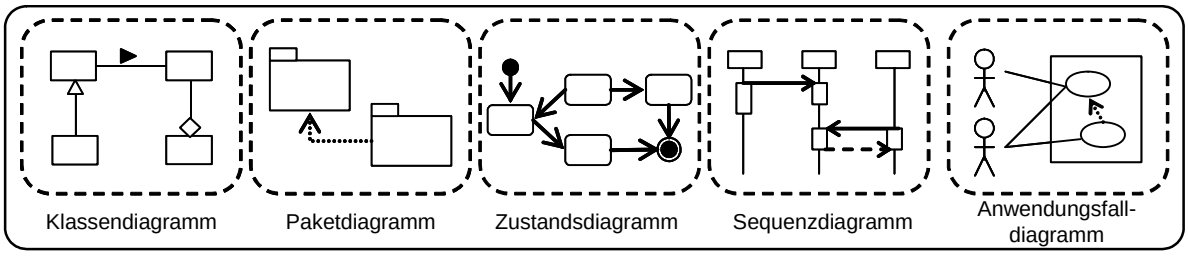


Abbildung 11-2: Diagrammtypen in der UML

Die UML definiert insgesamt 13 Diagramme, von denen sechs für die Strukturmodellierung und sieben für die Verhaltensbeschreibung gedacht sind. Es ist jedoch nicht erforderlich, sämtliche zur Verfügung stehenden Diagrammtypen auch einzusetzen. In der Praxis hat sich gezeigt, dass vor allem *Klassendiagramme*, *Zustandsdiagrammen*, *Sequenzdiagramme* und das *Anwendungsfalldiagramm* ausreichen, um ein System zu spezifizieren [RUMPE & SANDNER 2001]. Das Klassendiagramm legt fest, welche Klassen existieren und welche Be-

ziehungen zwischen ihnen bestehen. Es bildet die grundlegende Struktur eines Systems auf einer feingranularen Ebene ab. Die Klassen können zur besseren Übersicht in so genannte Pakete eingeordnet werden, deren Abhängigkeiten wiederum in Paketdiagrammen abzubilden sind. Die Funktionsabläufe und die Kommunikation der einzelnen Klassen können mit den Zustands- bzw. Sequenzdiagrammen dargestellt werden. Während der Zustandsautomat vorwiegend die möglichen Zustände eines Systems beschreibt, stellt das Sequenzdiagramm Nachrichten dar, die eine ausgewählte Menge von Objekten und Akteuren während einer bestimmten Zeitspanne miteinander austauschen. Der Anwendungsfall zeigt die Zusammenhänge zwischen bestimmten Anwendungsfällen und den daran beteiligten Akteuren und dient vorwiegend zur Aufnahme und Darstellung der Anforderungen. Eine ausführliche Beschreibung der UML ist in [JECKLE ET AL. 2004] zu finden.

Die UML ermöglicht es, zusätzliche, anwendungsspezifische Bedingungen und Integrationsregeln zu definieren. Diese so genannten *Constraints* beschränken die Semantik einzelner Modellelemente und müssen stets erfüllt sein. Constraints können mit unterschiedlichen Mitteln formuliert werden. Neben einer natürlich-sprachlichen Darstellung können beispielsweise *Stereotype* verwendet werden. Diese klassifizieren die mögliche Verwendung eines Modellelements, indem sie mehreren Modellelementen bestimmte gemeinsame Eigenschaften zuschreiben. Dabei kann auch die graphische Repräsentation der Modellelemente beeinflusst werden. Eine weitere Möglichkeit zur Definition von Constraints bietet die *Object Constraint Language* (OCL). Die OCL ist eine einfache formale Sprache, mit der den UML-Diagrammen weitere Semantik hinzugefügt werden kann. Der OCL-Formalismus basiert auf der Mengentheorie und erlaubt die Beschreibung von Zusicherungen sowie Vor- und Nachbedingungen in UML-Diagrammen [OESTEREICH 2005].

11.3 Modellierungssprachen für die Abbildung von Daten- und Produktstrukturen

11.3.1 Extensible Markup Language (XML)

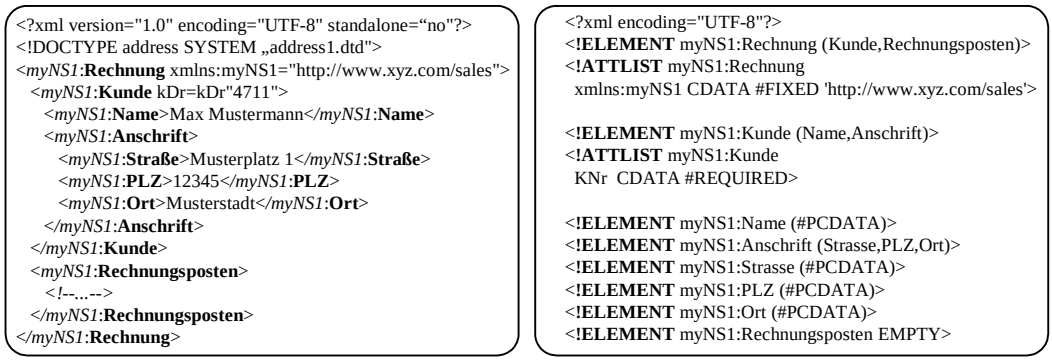
Die *Extensible Markup Language* (XML) [WORLD WIDE WEB CONSORTIUM 2000] ist eine Auszeichnungssprache. Damit können Textteile eines Dokuments mit bestimmten Anweisungen ausgezeichnet bzw. annotiert werden. Es gibt zwei Gruppen von Auszeichnungssprachen. Die XML gehört zur Gruppe der *Descriptive Markup Languages* (DML), welche die Syntax von Daten beschreiben. Die andere Gruppe der *Procedural Markup Languagse* (PML) spezifiziert hingegen die Schritte, die zur Darstellung der Daten notwendig sind. Vertreter dieser Gruppe sind unter anderem die Formate TeX, PDF und PostScript.

Grundelemente der XML

Die Anwendung der XML beschränkt sich mittlerweile nicht mehr nur auf die Auszeichnung von Textdokumenten [ALBRECHT ET AL. 2002]. Aufgrund der universellen Einsetzbarkeit liegt der Schwerpunkt heute vielmehr in der Definition von Datenformaten zum Informationsaustausch. XML wird unter anderem zur Beschreibung von elektrischen Betriebsmitteln, in der Materialwirtschaft und zur Anlagendokumentation eingesetzt ([WOLLSCHLAEGER 1999], [BUCHNER ET AL. 2000], [KIRMAIR & FAY 2001]). Zentrales Anliegen für den Einsatz von

XML ist es, Inhalte maschinell zugänglich, auffindbar und manipulierbar zu machen. Die XML bietet hierfür die Möglichkeit, Inhalte über kennzeichnende Markierungen in einzelne, funktionale Bereiche zu untergliedern. Diese Markierungen werden *Tags* genannt.

In Abbildung 11-3 ist ein Beispiel eines XML-Dokuments sowie die dazugehörige Dokumentgrammatik abgebildet. Es stellt Informationen über offene Rechnungen einzelner Kunden bereit.



a) XML-Dokument b) DTD-Dokumentgrammatik

Abbildung 11-3: Aufbau eines XML-Dokuments

Man unterscheidet in der XML zwischen zwei Grundtypen, der DTD-Dokumentgrammatik und dem eigentlichen XML-Dokument. Die *Document Type Definition* (DTD) beschreibt die Struktur eines Dokuments anhand einer definierten Grammatik. Es wird genau festgelegt, in welcher Reihenfolge die Tags in der XML-Datei auftreten und wie diese untereinander verschachtelt sein können. Eine DTD kann extern in einer Datei gespeichert sein oder im XML-Dokument selbst definiert werden.

Das XML-Dokument füllt die in der DTD definierte Gliederung der Daten mit entsprechenden Inhalten. Hier werden den Tags die Werte zugewiesen. Ein XML-Dokument kann wohlgeformt und/oder gültig sein. Wohlgeformt heißt, dass die in der XML-Spezifikation definierten Regeln korrekt angewandt worden sind. Gültigkeit setzt neben der Wohlgeformtheit auch die Existenz einer DTD voraus. Dabei muss die im XML-Dokument verwendete Struktur mit der aus der DTD übereinstimmen [W3C 2004A].

In der DTD werden die einzelnen Elemente, aus denen die XML-Dokumente aufgebaut sind, definiert. Mit *Element* wird dabei die Information bezeichnet, welche durch Tags umrahmt wird. Elemente sind die Basis des XML-Markup und stellen die logische Struktur eines XML-Dokuments dar. Im Beispiel aus Abbildung 11-3 b wird das Element *Rechnung* definiert, das aus den Elementen *Kunde* und *Rechnungsposten* besteht. Als Konsequenz aus dieser Festlegung werden im XML-Dokument die Informationen über den Kunden und dessen Rechnungsposten mit dem Start-Tag *<Rechnung>* und dem End-Tag *</Rechnung>* eingerahmt.

Des Weiteren können den Elementen Attribute zugewiesen werden. Darunter werden Schlüsselwort-Wert-Paare verstanden, die zusätzliche Informationen über Elemente beschreiben. Attribute werden einem Element in der DTD durch die Anweisung *<!ATTLIST…>* zugewiesen. Im Beispiel besitzt das Element *Kunde* das Attribut *KNr* vom Typ String. Zur Erläute-

rung weiterer Attributtypen sei auf die XML-Spezifikation [W3C 2004A] verwiesen. Bei der Definition von Attributen können noch weitere Vorgaben gemacht werden, beispielsweise um die Angabe eines Attributs im XML-Dokument explizit zu fordern oder dessen Wert fest vorzugeben.

Das Attribut *xmlns:myNS1* stellt eine Besonderheit dar. Es definiert einen Namensraum, um die Eindeutigkeit der Elemente der DTD gewährleisten zu können. Die Motivation für die Definition von Namensräumen liegt darin, dass jeder Fachbereich seine eigene Auszeichnungssprache entwickeln kann. Diese kann wiederum von anderen Fachbereichen verwendet werden, um beispielsweise die eigene Auszeichnungssprache zu erweitern. Bei der gleichzeitigen Anwendung mehrerer Sprachen kann es zu Konflikten kommen, wenn gleichnamige XML-Elemente verwendet werden. Namensräume lösen solche Mehrdeutigkeiten auf, indem sie vor jedes Element ein Präfix setzen, um anzuzeigen, zu welchem Fachbereich das Element gehört. Das Präfix wird dabei mit einem *Uniform Resource Identifier* (URI) verbunden. Ein URI ist eine Zeichenfolge, die zur Identifizierung einer abstrakten oder physikalischen Ressource dient.

Die DTD wurde vorwiegend für die Strukturbeschreibung von Text-Dokumenten entwickelt. Die Anwendungsbereiche der XML gehen jedoch weit darüber hinaus. Das Aufkommen von Schemasprachen Mitte der neunziger Jahre ist im Wesentlichen auf drei Anforderungen zurückzuführen, die mit den DTDs nicht umgesetzt werden können. Die größte Einschränkung betrifft das unzureichende Typkonzept. DTDs verfügen nur über ein beschränktes Inventar vordefinierter Datentypen. Es gibt beispielsweise keine Möglichkeit, festzulegen, dass ein Element zur Angabe eines Kalendermonats ein Integerwert zwischen eins und zwölf sein muss. Beim Datenaustausch zwischen mehreren Anwendungen obliegt es somit der Zielapplikation, die Gültigkeit der empfangenen Daten zu prüfen. Dadurch gehen die Vorteile der Flexibilität und Neutralität XML-basierter Datenformate verloren.

Ebenso ist es schwierig oder gar nicht möglich, neue Datentypen zu definieren [HAROLD 2004]. Dieser Umstand ist darauf zurückzuführen, dass DTDs nur eingeschränkt erweiterbar und skalierbar sind. Es ist schwierig, unabhängige DTDs miteinander zu kombinieren. Prinzipiell besteht die Möglichkeit, DTDs zu modularisieren. Dieses Modulkonzept basiert auf einem einfachen Textersetzungsmechanismus, der bei großen DTDs von beispielsweise 10.000 Codezeilen zu einem Verlust der Übersichtlichkeit führen kann.

Eine weitere Einschränkung beim Einsatz von DTDs liegt darin, dass sie in einer proprietären Notation kodiert werden. Eine DTD selbst ist keine XML-Sprache, was vor allem in der Praxis einen zusätzlichen Implementierungsaufwand zur Folge hat.

Schemasprachen beseitigen die aufgeführten Unzulänglichkeiten, indem sie ein Typkonzept verwenden, das sich an den Möglichkeiten moderner Programmiersprachen orientiert und leicht erweitert werden kann. Darüber hinaus verwenden alle Schemasprachen eine XML-Notation, so dass bei der Entwicklung und Verarbeitung von XML-Dokumenten und XML-Schemata dieselben Softwarewerkzeuge verwendet werden können. Von den zahlreichen Schemasprachen sind vor allem *XML-Schema* des *World Wide Web Consortiums (W3C)* [FALLSIDE & WALMSLEY 2004] [CAMPBELL ET AL. 2003] sowie *RELAX New Generation* (RELAX NG) der *Organization for the Advancement of Structured Information Standards*

(OASIS) [ISO/IEC JTC 1/SC 34 2002] einem größeren Anwenderkreis bekannt. Die XML-Schemasprache des W3C ist eine komplexe Sprache zur Beschreibung eines XML-Typsystems. Sie ermöglicht die Spezifikation neuer XML-Elemente sowie deren Attribute. Allerdings ist die Sprache sehr komplex und schwierig in der Anwendung. Die Schemasprache RELAX NG soll hier Abhilfe schaffen. Die Hauptziele bei deren Entwicklung waren Einfachheit und leichte Erlernbarkeit bei annähernd gleichem Funktionsumfang. Während der Fokus von XML-Schema auf der Definition der verwendeten Typen liegt [KLEVER 2001], stellt RELAX NG die Beschreibung der Dokumentstruktur in den Vordergrund. Die Spezifikation von RELAX NG definiert kein eigenes Typkonzept, sondern sieht vor, externe Typsysteme zu verwenden. So ist es möglich, die Struktur eines XML-Dokuments kompakt mit RELAX NG zu beschreiben und entsprechende Typdefinitionen aus XML-Schema zu verwenden [TILLY & TILKOV 2004].

Lebenszyklus von XML-Dokumenten

Bei der Verarbeitung von XML-Dokumenten nehmen so genannte *Parser* eine zentrale Rolle ein. Ein Parser ist eine Softwarekomponente, welche die Grammatik einer formalen Sprache akzeptiert. Auf der Basis der syntaktischen Regeln, die in der Grammatik hinterlegt sind, kann der Parser feststellen, ob eine Sprache von dieser Grammatik erzeugt wird oder nicht. In Abbildung 11-4 ist der Einsatz eines Parsers bei der Verarbeitung eines XML-Dokuments zu sehen.

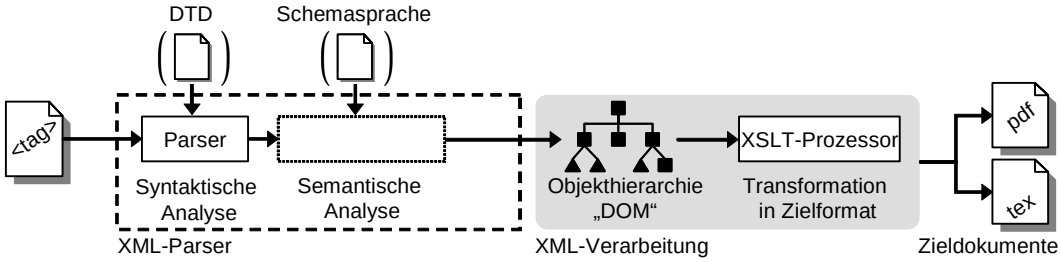


Abbildung 11-4: Prozessschritte bei der Verarbeitung eines XML-Dokuments [ALBRECHT & MEYER 2002]

Der Parser führt eine syntaktische Analyse der Dokumentstruktur durch, indem er die einzelnen Tags des Dokuments einliest und deren Reihung mit den Vorgaben der Grammatik prüft. Die Grammatik ist in einer DTD abgelegt, darüber hinaus können weitere Vorgaben in einem Schema hinterlegt werden. Die Anwendung einer DTD und eines Schemas auf ein und dasselbe Dokument ist erlaubt und in bestimmten Anwendungsfällen sinnvoll [HAROLD 2004]. Beim Einsatz einer Schemasprache kann bis zu einem gewissen Grad auch die Semantik des Dokuments überprüft werden, z. B. bei der Einhaltung der Wertebereiche bestimmter Attributdefinitionen. Parser, die DTDs und Schemasprachen verwenden, werden auch validierende XML-Parser genannt.

Der Parser überführt das XML-Dokument in eine abstrakte Baumstruktur, das *Document Object Model* (DOM). Die Knoten sind die XML-Elemente, die Kanten ergeben sich durch deren Verschachtelung. Das DOM definiert eine von der Programmierung unabhängige Menge abstrakter Schnittstellen zum Lesen und Schreiben von XML-Dokumenten. Es ist eine

API, die verschiedene Operationen zur Extraktion und Manipulation von Informationen aus dem XML-Dokument zur Verfügung stellt. In der Repräsentation des XML-Dokuments durch das DOM zeigt sich die Besonderheit des Dokument-Markups durch die XML. Die durch Tags annotierten Dokumentteile stehen in einem hierarchischen Zusammenhang, der sich als Baum abbilden lässt. Die Interpretation von Textdokumenten als Bäume hat einige Vorteile gegenüber einer linearen Dokumentauszeichnung. Wenn in einem Baum einem Element eine Eigenschaft zugeordnet ist, so kann z. B. festgelegt werden, dass diese Eigenschaft auch allen diesem Element untergeordneten Elementen zukommt, sofern bei ihnen nicht explizit etwas anderes vermerkt ist. Das hier angewendete Prinzip der Vererbung von Eigenschaften ist also ein Mittel, Teilen eines Dokuments diese Eigenschaften effizient zuzuordnen. Die Baumstruktur ermöglicht darüber hinaus eine effiziente Navigation durch die einzelnen XML-Elemente [LOBIN 2003]. Die W3C hat hierfür die Anfragesprache *XML Path Language (XPath)* entwickelt, mit der Teile eines XML-Dokuments adressiert werden können. So genannte XPath-Ausdrücke formulieren dabei Pfade durch die Baumstruktur der Dokumente, die um Eigenschaftsabfragen an die Attribute der XML-Elemente erweitert werden können [CLARK & DEROSE 1999]. *XQuery* ist eine weitere Möglichkeit, Datenanfragen an eine Menge von XML-Dokumenten zu formulieren. XQuery orientiert sich hierbei an den Sprachen für relationale Datenbanken [KEMPER & EICKLER 1997].

XPath stellt die Grundlage einer Reihe von weiteren XML-Standards dar, unter anderem auch der *XSL Transformation (XSLT)*, wobei XSL für *Extensible Stylesheet Language* steht. XSLT ist eine XML-Sprache zur Spezifikation von Regeln zur Transformation von XML-Dokumenten. Ein XSLT-Dokument ist aus einzelnen Templates aufgebaut. Ein Template besitzt ein aus XPath-Ausdrücken bestehendes Pattern. Dieses beschreibt die Knoten, auf die es angewendet werden soll, und besitzt einen Generierungsteil, der die Regeln für die Erzeugung des Zielformates enthält [HAROLD & MEANS 2004].

Grundlegender Aufbau von Dokumenten

XML-Dokumente besitzen eine logische und eine physikalische Struktur. Die logische Struktur ergibt sich durch die Abfolge der einzelnen XML-Elemente sowie durch deren Verschachtelung untereinander. Ein Dokument kann eine XML-Deklaration besitzen, die Daten über die verwendete XML-Version oder Zeichenkodierung enthält (siehe Abbildung 11-3). Des Weiteren kann im Deklarationsteil noch eine DTD oder eine Schema-Spezifikation angegeben werden, gegen die der Parser das Dokument prüfen soll. Dem Deklarationsteil folgt das Wurzelelement, das alle anderen Elemente des Dokuments einschließt.

Ein einzelnes XML-Dokument kann Daten und Element-Deklarationen aus vielen verschiedenen Quellen und vielen unterschiedlichen Dateien beziehen. Die Speicherungseinheiten, die bestimmte Teile eines XML-Dokuments enthalten, werden *Entitäten* genannt [HAROLD 2004]. Sie sind die Atome der XML und bilden die physikalische Struktur eines XML-Dokuments ab. Entitäten korrespondieren meist mit Dateien, wobei aber prinzipiell weitere Ressourcen wie Datenbanken möglich sind [BOX ET AL. 2001]. Die Speichereinheit, welche die XML-Deklaration, die DTD-Deklaration und das Wurzelelement enthält, wird als Dokument-Entität bezeichnet. Das Wurzelelement und dessen XML-Elemente enthalten Entitätsreferenzen, die auf weitere Daten verweisen, die in das Dokument eingefügt werden sollen. Ein XML-Parser kombiniert alle verschiedenen referenzierten Entitäten zu einem einzigen logischen Doku-

ment. Die meisten Entitäten haben Namen, über die sie referenziert werden können.

11.3.2 Die Datenmodellierungssprache EXPRESS

EXPRESS (Expressive Power) ist eine objektorientierte Datendefinitionssprache. Sie wird in der Serie 11 der ISO 10303 Normenreihe definiert. Alle Datenmodelle in STEP, sowohl in den Informationsmodellen als auch in den Anwendungsprotokollen, werden mit der Sprache EXPRESS beschrieben. Eine wesentliche Anforderung bei der Entwicklung dieser Sprache lag darin, dass sie für den Menschen lesbar und für den Computer verarbeitbar war. Im Folgenden werden die grundlegenden Sprachelemente eingeführt. Eine detaillierte Beschreibung ist in [ISO 10303-11 2004] zu finden.

Schemata

Ein Schema dient dazu, bestimmte Interessensbereiche abzugrenzen und das gesamte Datenmodell zu strukturieren und zu gliedern. Ein Schema definiert einen eigenen Namensraum für alle enthaltenen Komponenten, wobei Referenzen zwischen einzelnen *Schemata* möglich sind. Eine Schema-Spezifikation wird durch das Schlüsselwort *Schema* eingeleitet. Anschließend können Entitäten, Typen, Regeln und Algorithmen deklariert werden, wobei die Reihenfolge keinen Einfluss auf die Semantik hat.

Typen und Entitäten

Diese beiden Sprachelemente bilden die Grundlage für die Entwicklung von Datenstrukturen. Mit ihnen können Daten und deren Beziehungen untereinander abgebildet werden. EXPRESS stellt verschiedene Typen und Typkonstruktionen zur Verfügung, unter anderem Basistypen (*Simple Types*) wie INTEGER, REAL oder STRING sowie Aufzählungstypen. Ein Beispiel hierfür ist in Abbildung 11-5 zu sehen. Der Typ *Geschlecht* kann die Werte *weiblich* oder *männlich* annehmen.

```
SCHEMA Beispiel_A;
  REFERENCE FROM Beispiel_B(Person);
  ENTITY Mann SUBTYPE OF Person;
  DERIVE
    SELF\Geschlecht : Geschlecht := maennlich;
  INVERSE
    Ehemann_von : SET [0:1] OF Frau FOR Ehefrau_von;
  END_ENTITY;
  ENTITY Frau SUBTYPE OF Person;
    Ehefrau_von : OPTIONAL Mann;
  DERIVE
    SELF\Geschlecht : Geschlecht := weiblich;
  END_ENTITY;
END_SCHEMA;
```

```
SCHEMA Beispiel_B;
  TYPE Geschlecht = (weiblich, männlich);
  ENTITY Person SUPERTYPE OF ONEOF (Frau, Mann);
    Name, Vorname : STRING;
    Geburtsjahr : INTEGER;
    Geschlecht : Geschlecht;
  END_ENTITY;
END_SCHEMA;
```

Abbildung 11-5: Beispiel in der Datenmodellierungssprache EXPRESS

Entitäten können mit Klassen aus der objektorientierten Programmierung verglichen werden. Die Deklaration beginnt mit dem Schlüsselwort ENTITY. Anschließend folgen der Name der Entität, die Angabe eventueller Vererbungsbeziehungen zu Super- und Subtypen, die Angabe

von Attributen sowie die Definition von Regeln. Durch die Vererbung enthalten die Subklassen alle Bestandteile ihrer Superklassen. Der Namensraum einer Entiät setzt sich aus dem eigenen und denen aller Superklassen zusammen. Das Schlüsselwort *SELF* stellt eine Art Zeiger auf die eigene Entität dar und wird dann eingesetzt, wenn lokale und vererbte Attribute voneinander unterschieden werden sollen. Ohne diese spezielle Kennzeichnung wären das lokal deklarierte und das vererbte Attribut der Entität sichtbar, obwohl sie beide dieselbe Bezeichnung besitzen.

Eine Besonderheit von EXPRESS ist die Möglichkeit, Vererbungsbeziehungen in drei verschiedenen Formen angeben zu können. Im Beispiel aus Abbildung 11-5 ist die Entität *Person* als Supertyp der Entität *Mann* und *Frau* mit dem Statement *SUPERTYPE OF ONEOF* deklariert. Dies bedeutet, dass eine Person entweder eine Frau oder ein Mann sein kann. Darüber hinaus kann mit dem Schlüsselwort ANDOR angegeben werden, dass eine Person ein Mann, eine Frau oder beides sein kann. Das Schlüsselwort AND legt fest, dass eine Person immer ein Mann und eine Frau ist. Bei keiner detaillierten Angabe wird automatisch eine ANDOR-Vererbung benutzt.

Wird in einer Entität ein Attribut vom Typ einer Entität angegeben, so wird dadurch eine Referenz zwischen diesen beiden Entitäts-Typen hergestellt. Im Beispiel ist eine 1:1-Beziehung zwischen Mann und Frau spezifiziert. Um die Konsistenzerhaltung sicherzustellen, kann zu jeder Referenz auch eine Gegenreferenz in Form von inversen Attributen formuliert werden. Eine Gegenreferenz wird mit dem Schlüsselwort INVERSE eingeleitet und stellt einen Mechanismus für die Überwachung der referenziellen Integrität in Form einer lokalen Regel dar.

Referenzen auf Entity-Typen verschiedener Schemata sind durch die Schlüsselwörter *USE* und *REFERENCE* möglich. Damit werden die Entitäten aus den anderen Schemata in den Namensraum des lokalen Schemas eingefügt. Bei der Referenz mit USE lassen sich die importierten Entitäten so verwenden, als ob sie lokal definiert worden wären. Insbesondere können sie wieder von einem dritten Schema importiert werden. Mit REFERENCE importierte Entitäten können hingegen nur innerhalb des entsprechenden Schemas verwendet werden.

EXPRESS ist eine rein textuelle Sprache. Komplexe Datenmodelle lassen sich damit kompakt darstellen und effizient verarbeiten. Ein schneller Überblick über die dargestellten Zusammenhänge ist damit aber nicht möglich. Zu diesem Zweck wurde die Sprache *EXPRESS-G* entwickelt. In Abbildung 11-6 sind einige der grundlegenden Symbole zu sehen.

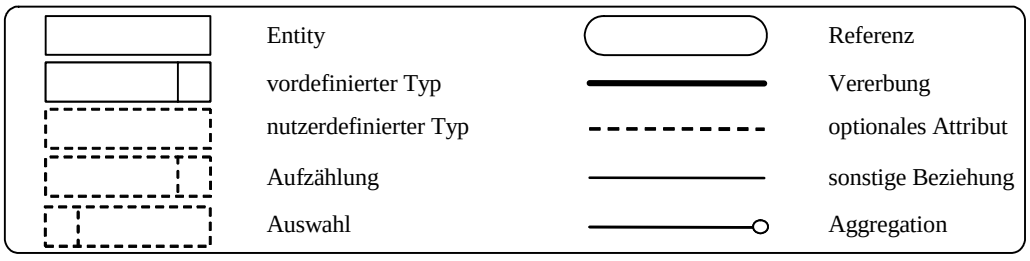


Abbildung 11-6: Symbole der Sprache EXPRESS-G

EXPRESS-G ist eine Untermenge von EXPRESS. Daher können nicht alle EXPRESS-Datenmodelle in eine graphische Form der EXPRESS-G transformiert werden.

11.3.3 ISO 10303 Part 21: Clear Text Encoding of the Exchange Structure

Die Serie 21 der ISO 10303 gehört zur Klasse der Implementationsmethoden. Dieser Teil der Norm legt ein Format für die Austauschstruktur von Datenmodellen fest, die in der Sprache EXPRESS beschrieben worden sind. Im Wesentlichen handelt es sich dabei um eine ASCII-Datei, die sich in zwei Bereiche gliedert. Die *HEADER SECTION* enthält Informationen zur Datei selbst, wie beispielsweise den Namen, das Datum, den Autor und mit welchem Softwarewerkzeug die Datei erstellt wurde. In der *DATA SECTION* sind die Informationen zum Datenmodell selbst enthalten (siehe Abbildung 11-7).

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(
/* description */ ('A minimal AP214 example with a single part'),
/* implementation_level */ '2;1');
FILE_NAME(
/* name */ 'demo',
/* time_stamp */ '2003-12-27T11:57:53',
/* author */ ('Lothar Klein'),
/* organization */ ('LKSoft'),
/* preprocessor_version */ ' ',
/* originating_system */ 'IDA-STEP',
/* authorization */ ' ');
FILE_SCHEMA (('AUTOMOTIVE_DESIGN { 1 0 10303 214 2 1 1}'))
ENDSEC;
DATA;
#10=ORGANIZATION('O0001','LKSoft','company');
#11=PRODUCT_DEFINITION_CONTEXT('part definition',#12,'manufacturing');
#12=APPLICATION_CONTEXT('mechanical design');
#13=APPLICATION_PROTOCOL_DEFINITION(,'automotive_design',2003,#12);
#14=PRODUCT_DEFINITION('0',$,#15,#11);
#15=PRODUCT_DEFINITION_FORMATION('1',$,#16);
#16=PRODUCT('A0001','Test Part 1',,(#18));
#17=PRODUCT_RELATED_PRODUCT_CATEGORY('part',$,(#16));
#18=PRODUCT_CONTEXT(,#12,);
#19=APPLIED_ORGANIZATION_ASSIGNMENT(#10,#20,(#16));
#20=ORGANIZATION_ROLE('id owner');
ENDSEC;
END-ISO-10303-21;
```

Abbildung 11-7: STEP-Datei nach ISO 10303-21

Die *HEADER* Section enthält immer je eine Instanz der Entität *FILE_DESCRIPTION*, *FILE_NAME* und *FILE_SCHEMA*. In der FILE_DESCRIPTION kann eine allgemeine Beschreibung zum Dateiinhalt angegeben werden. Außerdem wird der so genannte *Implementationlevel* angegeben. Die Attribute von *FILE_NAME* sind der Name der Datei, der Zeitpunkt ihres Erzeugens, die verantwortliche Person, die Organisation, zu welcher diese gehört, das System, mit dessen Hilfe die Datei erzeugt wurde, sowie das Ursprungssystem und der Verantwortliche für eine Übernahme aus diesem System. Im Normalfall werden nicht alle Attribute mit Werten belegt, speziell in einer Testphase beschränkt man sich meist auf den Namen, den Zeitpunkt und das Erzeugersystem. FILE_SCHEMA hat nur ein Attribut: eine Liste der Namen derjenigen Schemata, deren Entitäten in dieser Datei instanziiert sind. Die *DATA*

SECTION enthält die zu speichernden bzw. auszutauschenden Daten. Jede Instanz einer Entität darf höchstens einmal vorkommen. Die Abbildung in der Datei erfolgt durch das Nummerzeichen '#', gefolgt von einer natürlichen Zahl, dem Gleichheitszeichen '=' und dem Namen der zugehörigen Entitätsdefinition. In runden Klammern werden dann die Werte der Attribute in der Reihenfolge, die derjenigen in der Entitätsdefinition entspricht, angegeben. Referenzen zu anderen Entitäten werden durch Angabe der entsprechenden Nummern kenntlich gemacht, Zeichenketten müssen in einfache Anführungszeichen, Werte aus Aufzählungen (Enumerationen) in Punkte gesetzt werden. Listen werden durch runde Klammern geöffnet und geschlossen. Anstelle nicht belegter optionaler Attribute wird ein Dollarzeichen '$' angegeben. Eine Datei im Part 21-Format wird auch als *STEP physical exchange file* bezeichnet.

11.4 Die Programmiersprachen der IEC 61131-3

Für die Programmierung speicherprogrammierbarer Steuerungen sind zahlreiche Sprachen entwickelt worden, die ihren Ursprung in der Schalttechnik oder der Mikroprogrammierung haben [Sabbah 2000]. Die International Electrotechnical Commission (IEC) fasste 1994 die in den letzten 20 Jahren gesammelten Erfahrungen im SPS-Bereich zusammen und entwickelte den Standard IEC 61131-3 *Programmable Controllers*. In der IEC 61131-3 wurden insgesamt fünf Programmiersprachen definiert. Das Ziel war es, die bis dahin verwendeten Programmiersprachen zu vereinheitlichen und dadurch unabhängig von proprietärer Steuerungshardware sowie Programmiersystemen zu werden [Pühl 1999]. Wie in Abbildung 11-8 zu sehen, wurden zwei textorientierte und drei graphische Programmiersprachen standardisiert.

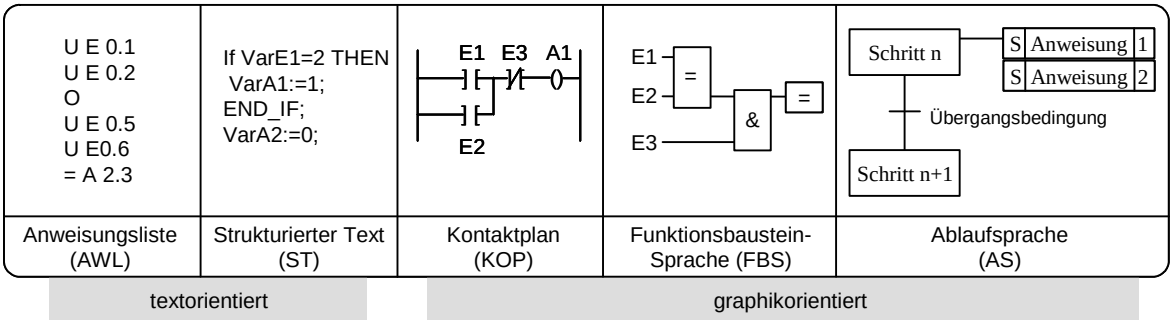


Abbildung 11-8: Programmiersprachen der IEC 61131-3

Die Anweisungsliste (engl. *Instruction List*, IL) ist eine maschinennahe Programmiersprache. Sie ist vergleichbar mit einer Assemblersprache und sie ist universell einsetzbar. Der Strukturierte Text (engl. *Structured Text*, ST) hingegen gehört zur Familie der höheren Programmiersprachen wie beispielsweise C++ oder Java und bietet auch vergleichbare Sprachelemente wie Schleifenkonstrukte und bedingte Abfragen. Der Kontaktplan stammt ursprünglich aus dem Bereich der elektromechanischen Schützsteuerungen. Das Systemverhalten wird dabei durch die Darstellung des Stromflusses durch einzelne Netzwerke beschrieben. An den beiden Seiten des Kontaktplans befinden sich so genannte Stromschienen. Durch die linke Schiene fließt Strom und sie besitzt dadurch den Zustand 1. Zwischen den beiden Schienen führen Verbindungen den Strom zu Variablen, die abhängig von ihrem logischen Zustand eine Weiterführung ermöglichen oder verhindern. Diese Variablen können hintereinander oder parallel verknüpft sein. Die Funktionsbausteinsprache kommt aus der Signalverarbeitung. Ähnlich wie

beim Kontaktplan werden auch hier die Anweisungen in einzelne Netzwerke gegliedert. Diese werden als logische Schaltpläne mit Logikbausteinen (UND-Gatter, ODER-Gatter) dargestellt und können um zusätzliche Bausteine wie Flip-Flops, Zähler oder Taktgeber erweitert werden. Die Ablaufsprache (engl. *Sequential Function Chart*, SFC) ist eine Weiterentwicklung der Schrittkettenprogrammierung. Sie bietet bessere Möglichkeiten, komplexe Aufgaben in überschaubare Einheiten zu zerlegen und den Informationsfluss zwischen diesen Einheiten abzubilden. Die Ablaufsprache besteht aus Schritten und aus Transitionen, welche die einzelnen Schritte miteinander verbinden. Schritte können aktiv oder inaktiv sein. Schritte sind mit Aktionsblöcken verknüpft, die ausgeführt werden, solange der Schritt aktiv ist. Ein Schritt wird inaktiv, wenn die Bedingung der Ausgangstransition erfüllt ist.

Mit der Ablaufsprache können sequentielle, alternative, iterative und parallel ablaufende Prozesse abgebildet werden.

Die IEC 61131-3 definiert insgesamt drei Programmorganisationseinheiten (POE). Die *Funktion* ist eine parametrierbare POE, die keine eigenen statischen Variablen besitzt. Bei gleichen Eingangswerten liefert eine Funktion stets denselben Ergebniswert zurück. Der *Funktionsbaustein* (FB) ist ebenfalls parametrierbar und besitzt im Gegensatz zur Funktion lokale Variablen. Der Ergebniswert hängt damit von den Eingangswerten und dem Zustand der lokalen Variablen ab. Das *Programm* koordiniert den Aufruf aller anderen POE.

11.5 Übersetzung englischer Zitate

Die anschließende Tabelle zeigt die in dieser Arbeit enthaltenen englischen Textstellen und deren Übersetzung in die deutsche Sprache.

	Zitat aus [GRUBER 1993] (siehe Seite 11)
en	An ontology is an explicit specification of a conceptualization.
de	Eine Ontologie ist eine explizite formale Spezifikation einer gemeinsamen Konzeptualisierung (Begriffsbildung).
	Zitat aus [WIEDERHOLD 1992] (siehe Seite 13)
en	Mediation simplifies abstracts, reduces, merges and explains data. […] Mediation covers a wide variety of functions that enhance stored data prior to their use in an application. Media-tion makes an interface intelligent by dealing with representation and abstraction problems that you must face when trying to […] use data and knowledge resources.
de	Die Funktion "Mediation" vereinfacht, abstrahiert, reduziert, mischt und erklärt Daten. Die Mediation umfasst eine große Menge an Funktionen, welche die Daten vor deren Anwendung in den Applikationen aufbereitet. Sie macht eine Schnittstelle intelligent, indem sie Daten geeignet repräsentiert und abstrahiert. Diese beiden Punkte sind für die Verwendung von Daten- und Wissensquellen von grundlegender Bedeutung.

Tabelle 11-1: Übersetzung englischer Zitate in die deutsche Sprache

11.6 Verwendete Softwarewerkzeuge

Im Folgenden sind alle Softwaresysteme bzw. Softwarekomponenten aufgeführt, die bei der Umsetzung des Metadatenen-Management-Systems eingesetzt wurden.

Name	Version	Hersteller	Beschreibung
Pro/Engineer Wildfire	2.0	Parametric Technology Corporation (PTC) http://www.ptc.com	3D-CAD
Catia	V5	Dassault Systèmes http://www.3ds.com	3D-CAD
Doors	6.1	Telelogic Deutschland GmbH http://www.telelogic.de	Anforderungs-management
Together	6.1	Borland GmbH http://www.borland.de	CASE-Tool
Open Office	2.0	Sun Microsystems Inc. Http://www.openoffice.org	Office-Paket
Poseidon	1.6.1	Gentleware AG http://www.gentleware.com/	CASE-Tool
eXist	1.0	Wolfgang Meier http://exist-db.org/	XML-Datenbank
Eclipse	3.1	Eclipse Foundation Inc. Http://www.eclipse.org	Entwicklungs-umgebung Java
java	1.5	Sun Microsystems Inc. http://java.sun.com/	Programmier-sprache
javacc	3.2	Sun Microsystems Inc. https://javacc.dev.java.net/	Parser-Generator
jdom	1.0	Jason Hunter http://www.jdom.org	Bibliothek für die XML-Verarbeitung
jing	20030619	Thai Open Source Software Center Ltd http://www.thaiopensource.com	Validator für Relax-NG
trang	20030619	Thai Open Source Software Center Ltd http://www.thaiopensource.com	Transformator für Relax-NG

Tabelle 11-2: Verwendete Softwarewerkzeuge