Ein Konzept für die Datenhaltung in der fertigungsnahen Mikrosystemtechnik

Vom Fachbereich Elektrotechnik und Informatik der Universität Siegen

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

Genehmigte Dissertation

von

Dipl.-Inf. Jens Popp

1. Gutachter: Prof. Dr. Rainer Brück
2. Gutachter: Prof. Dr. Dietmar Fey
Vorsitzender: Prof. Dr. Madjid Fathi-Torbaghan

Tag der mündlichen Prüfung: 10.02.2005
urn:nbn:de:hbz:467-1009

Abstract

This work presents results generated during the PRINCE project. PRINCE is an acronym for PRocess INformation and management CEnter. The aim of the project was the support of the design of microsystems with special attention to the technology driven aspects of the design. The technology driven design (design for manufacturing) support is until now only marginally available.

Within the project a systematic analysis of the technology driven design was performed. It turns out that a lot of data is accumulated. Also most of the collected data is somehow related. Therefore the organized management of data is a key topic in the design of microsystems.

Starting from this realization methods for data management were researched for adequacy. Among the methods known in computer science the relational database was selected. As a result a relational database scheme is presented. All collected data can be stored and related in a useful manner using this scheme. Furthermore an extension of this scheme is presented that enhances it with methods from object oriented modelling. The usage of inheritance and multiple inheritance is discussed in detail.

With the PRINCE software a system is presented that puts the model into practical use. Using modern technologies like Java and J2EE the software enables collaborative work among users with different hardware platforms.

Inhaltsverzeichnis

1 Einleitung

Die Mikrosystemtechnik hat sich in den letzten Jahren zu einer der Zukunftstechnologien entwickelt. Viele Produkte unseres täglichen Lebens beinhalten bereits mikrosystemtechnisch gefertigte Teile. Ob in der Automobil- oder Medizintechnik, in der Luft- und Raumfahrtechnik - überall sind Anwendungen der Mikrosystemtechnik zu finden.

Gerade in den sicherheitskritischen Bereichen, wie zum Beispiel in den Auslösemechanismen eines Airbag-Systems, müssen die entsprechenden Produkte dabei höchsten Anforderungen genügen. Umfangreiche Vorstudien und Tests der Systeme verursachen hier Entwicklungszeiten von mehreren Jahren bis hin zu mehr als einem Jahrzehnt für ein neues Produkt.

Besonders die Erstellung der Anweisungen zur Herstellung neuer Produkte erfordert sehr viel Zeit. Hier müssen Maschinen getestet und angepasst, Materialien auf ihre Eigenschaften untersucht und neue Funktionsprinzipien erforscht werden. Dabei geht viel Zeit verloren durch die Wiederholung von Versuchen, die bereits mehrmals von anderen Mitarbeitern durchgeführt wurden. Durch die hohe Fluktuation der Mitarbeiter und die Spezialisierung auf enge Teilbereiche wird das erworbene Wissen oft nicht weitergegeben und somit auch nicht effizient eingesetzt.

Eine Ursache hierfür liegt an der ungenügenden Dokumentation. In der Industrie kommen bei der Entwicklung neuer Fertigungsprozesse die unterschiedlichsten Mittel zur Dokumentation zum Einsatz. Vom aus Papier hergestellten Laborbuch bis hin zum Einsatz von selbst erstellten Datenbanken ist alles zu finden. Skizzen und Zeichnungen werden mit Grafikprogrammen angefertigt, Prozessschritte in Office-Dokumenten beschrieben. All diese Daten befinden sich dann oft auf einem gemeinsamen File-Server, auf dem sie unter mehr oder minder kryptischen Dateinamen meist in Ruhe altern.

Ein weiteres Problem, welches hohe Entwicklungszeiten verursacht, ist das Fehlen von Entwurfswerkzeugen, die einen durchgängigen Entwurf ermöglichen. Heute sind für nahezu jede Stufe im Entwurfsprozess eigenständige Softwarelösungen im Einsatz. Die Ergebnisse einer Prozesssimulation müssen dann beispielsweise von Hand in die Simulation des Verhaltens eingebracht werden. Eine gemeinsamen Schnittstelle und die Durchgängigkeit der Unterstützung des Entwurfs sind auch unter den meist genannten Forderungen einer Studie der ZVEI [LN04] über den Einsatz von und den Bedarf an Entwurfswerkzeugen in der Mikrosystemtechnik.

Die Untersuchung eben dieser Problemstellungen ist einer der Themenbereiche, mit dem sich die Fachgruppe Mikrosystemetnwurf seit mehreren Jahren beschäftigt. Nach vorherigen Projekten, wie LIDO und Transtec (siehe Kapitel 3), wurden das Forschungsprojekt PRINCE in Zusammenarbeit mit der Robert Bosch GmbH in Gerlingen gestartet. Ziel des Projektes war es, aktuelle Software-Technologien zu nutzen, um den fertigungsnahen Entwurf von Mikrosystemen zu unterstützen.

Zur Vorbereitung wurden im Rahmen des Projektes intensive Gespräche mit den künftigen Anwendern der Software geführt. Dabei stellte sich schnell heraus, dass die Verwaltung der anfallenden Daten eines der Hauptprobleme darstellt. Ohne die rechnergerechte Aufbereitung dieser Daten war es nicht möglich eine Software zu entwickeln, die den Entwurfsprozess angemessen unterstützt.

Die Ergebnisse der Untersuchung, wie die Aufbereitung und Speicherung der Daten stattfinden kann, werden in dieser Arbeit vorgestellt. Eine erste Umsetzung der vorgestellten Konzepte wird mit dem Software System PRINCE eingeführt. Außerdem werden auch erste Erkenntnisse aus dem aktuellen EU-Projekt Promenade [Eur04], an dem das Institut beteiligt ist, präsentiert.

2 Mikrosystemtechnik

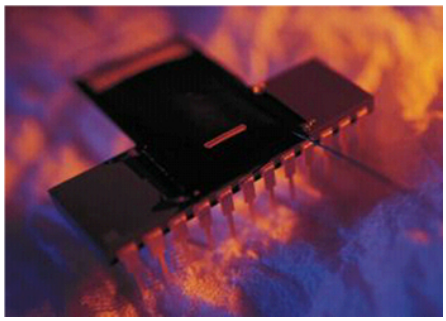
Der Begriff Mikrosystemtechnik setzt sich aus den Teilen Mikro, System und Technik zusammen. Die erste Silbe bezieht sich auf die Ausmaße der verwendeten Bauteile. In der Mikrosystemtechnik werden Strukturgrößen, die sich im Mikrometer-Bereich bewegen, erreicht. Der zweite Teil - System - soll deutlich machen, dass komplexe Systeme, die aus mehreren Komponenten bestehen können, hergestellt werden. Schließlich bedeutet der Teil Technik, dass technische Verfahren zur Herstellung solcher Mikrosysteme existieren und angewandt werden.

Der bekannteste Anwendungsbereich für Produkte aus der Mikrosystemtechnik ist die Automobilindustrie. Begriffe wie ESP[1] und Airbag sind heute jedem Autokäufer bekannt. Hinter diesen Technologien stehen Sensoren aus der Mikrosystemtechnik.

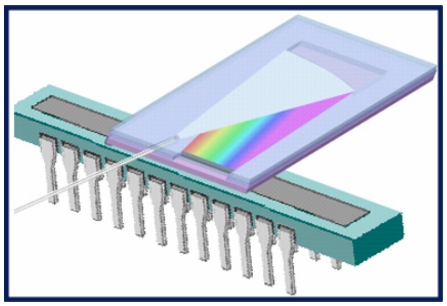
Aber auch in anderen Bereichen werden Produkte aus der Mikrosystemtechnik erfolgreich angewandt. Abbildung 2.1(a) zeigt ein Mikrospektrometer, das mit dem Liga-Verfahren (siehe Abschnitt 2.2.2) hergestellt wurde. Dabei wird ein eingespeistes Lichtsignal in seine Farbbestandteile zerlegt und von Sensoren ausgewertet. Dies ist in Abbildung 2.1(b) zu sehen. Dieses Spektrometer wird zum Beispiel in der Medizintechnik angewandt. Mit Bili-Check der Firma steag microParts [STE04](siehe Abbildung 2.1(c)) steht ein Produkt zur Verfügung, mit dem der Bilirubin-Wert bei Neugeborenen gemessen werden kann. Durch die Anwendung des handlichen Gerätes können Kosten gespart und ein frühzeitige Behandlung begonnen werden.

Einer der Hauptmerkmale der Mikrosystemtechnik ist die Verwendung vielfältiger Herstellungsverfahren und Materialien. Das Spektrum reicht von dem aus der Mikroelektronik bekannten Verfahren und Silizium als Basismaterial

[1] Elektronisches Stabilitätsprogramm



(a) Mikrospektrometer



(b) Schematischer Aufbau

BiliCheck

(c) Bill-Check

Abbildung 2.1: Verwendung eines Mikrospektrometers in der Medizintechnik

bis hin zu feinwerktechnischen und Laser-Verfahren, die mit Polymeren und Keramiken arbeiten. Dieses Kapitel soll anhand einiger Beispiele die Komplexität der verwendeten Technologien verdeutlichen.

2.1 Materialien

Unter den in der Mikrosystemtechnik verwendeten Materialien nehmen Silizium und dessen Verbindungen immer noch eine dominante Rolle ein. Dies lässt sich damit erklären, dass diese Materialien seit Jahren bekannt und im Einsatz sind. Außerdem besitzt Silizium hervorragende mechanische Eigenschaften, die es für den Einsatz in bestimmten Domänen prädestinieren.

Tabelle 2.1 zeigt die Eingeschaften von Silizium im Vergleich mit einigen anderen Werkstoffen. Dabei ist zu beachten, dass sich die Eigenschaften der Materialien je nach Verfahren, mit dem sie erzeugt wurden, unterschiedlich sind. Das lässt sich dadurch erklären, dass die entstehenden Materialien unterschiedliche Gitterstrukturen und Einschlüsse erhalten. Diese beeinflussen die gemessenen Werte.

An den Werten in der Tabelle ist zu sehen, dass Silizium in Härte, Bruchfestigkeit und Elastizität durchaus mit Stahl vergleichbar beziehungsweise sogar besser als Stahl ist. Damit ist Silizium hervorragend zur Herstellung mechanischer Strukturen geeignet. Aber auch die Siliziumverbindungen, wie Si_3N_4 oder SiO_2, finden in der Mikrosystemtechnik Anwendung. Als Konstruktionswerkstoffe finden sie Anwendung zur Maskierung, Passivierung oder als Opferschicht zur Herstellung freitragender Strukturen.

Neben den Materialien, die hauptsächlich die Struktur bestimmen, gibt es auch Funktionswerkstoffe. Diese Werkstoffe tragen direkt durch ihre Eigenschaften zur Wandlung der Signale bei. Als Beispiel hierfür kann die Nutzung des piezoelektrischen Effektes bei ZnO_2 angesehen werden [Sch04a].

Natürlich werden neben den aus der Mikroelektronik bekannten Werkstoffen auch andere Materialien eingesetzt. Dies ist insbesondere der Fall, wenn neue oder veränderte Fertigungsverfahren verwendet werden. Einige Beispiele hierfür sind die Verwendung von Metallen in der Feinwerktechnik und Keramiken und Polymeren in der Liga-Technik.

Werkstoff	Bruch-festig-keit $(10^9 \frac{N}{m^2})$	Knoop-Härte $(\frac{kg}{mm^2})$	Elasti-zitäts-modul $(10^1 1 \frac{N}{m^2})$	Dichte $(\frac{g}{cm^2})$	Therm. Leit-fähig-keit $(\frac{W}{cm}K)$	Therm. Ausdehnungs-koeffizient $(10^{-6}K^{-1})$
Diamant	53.0	7000	10.35	3.5	20.00	1.00
rostf. Stahl	2.1	660	2.00	7.9	0.33	17.30
SiC	21.0	2480	7.00	3.2	3.50	3.30
Al_2O_3	15.4	2100	5.30	4.0	0.50	5.40
Si_3N_4	14.0	3486	3.85	3.1	0.19	0.80
Fe	12.6	400	1.96	7.8	0.80	12.00
SiO_2	8.4	820	0.73	2.5	0.01	0.55
Al	0.2	130	0.70	2.7	2.36	25.00
Si	7.0	850	1.70	2.3	1.57	2.33

Tabelle 2.1: Mechanische und thermische Eigenschaften von Si im Vergleich zu anderen Werkstoffen [Mes00].

2.2 Fertigungsprozesse

Nicht nur die Materialien sondern auch die Fertigungsverfahren bestimmen Funktion und Struktur eines Mikrosystems. Zur Herstellung können die verschiedensten Verfahren zum Einsatz kommen. Die Spanne reicht dabei von den aus der Mikroelektronik bekannten Verfahren zur Bearbeitung von Siliziumwafern bis hin zu aus der Mechanik bekannten Verfahren wie Bohren und Fräsen.

2.2.1 Prozesse aus der siliziumbasierten Mikrosystemtechnik

Der Entwurf und die Fertigung von Mikrochips ist an sich äußerst kostspielig. Der Erfolg der Mikroelektronik beruht darauf, dass diese Kosten durch Wiederholung ähnlicher Prozessschritte und die Produktion im Batchpbetrieb auf viele Produkte aufgeteilt werden können. Werden diese Prozesse für die Fertigung von Mikrosystemen genutzt, so spricht man von der Oberflächen-Mikromechanik. Durch Einhaltung einiger Restriktionen kann dabei sicherge-

stellt werden, dass die gefertigten mechanischen Strukturen mit CMOS-Mikroelektronik kompatibel bleiben. So kann auf einem Chip die Mechanik, Sensorik und Auswerteelektronik vereinigt werden.

Im folgenden sollen die wichtigsten Gruppen von in der Oberflächen-Mikromechanik eingesetzten Prozessschritten vorgestellt werden. Dabei ist zu beachten, dass es noch viele weitere Verfahren und Abwandlungen der vorgestellten Verfahren gibt. Genauere Angaben über diese können der Fachliteratur, wie zum Beispiel [Mad02, FR97], entnommen werden.

Schichtabscheidung

Mit Verfahren zur Schichtabscheidung können neue Materialschichten auf dem Wafer erzeugt werden. Hierzu können verschiedene Verfahren eingesetzt werden. Die wichtigsten Verfahren sind:

PVD ist die Abkürzung für Physical Vapor Deposition. Hier werden für die Abscheidung physikalische Verfahren verwendet. Typische Beispiele für PVD Verfahren sind Aufdampfen und Sputtern.

CVD steht für Chemical Vapor Deposition. Bei CVD Verfahren werden die Schichten durch chemische Reaktionen direkt aus der Gasphase abgeschieden. Durch Veränderung der Umgebungsparameter kann sowohl die Geschwindigkeit der Abscheidung als auch die Struktur des abgeschiedenen Materials beeinflusst werden. Unterschieden werden unter anderem PECVD (Plasma Enhanced CVD), LPCVD (Low Pressure CVD) und Epitaxie. Mit der Epitaxie können Materialien aufgetragen werden, die die Gitterstruktur des Untergrundes fortführen.

Spin-On Verfahren dienen primär zum Aufbringen von Polymeren. Der Wafer wird dabei in Rotation versetzt und das Polymer in flüssiger Form aufgetropft. Durch die Rotation entsteht ein dünner Film des Polymers auf der Oberfläche.

Das verwendete Verfahren hat dabei nicht nur Einfluss auf die erzeugte Schicht sondern kann auch darunter liegende Schichten verändern. So können

beim Abscheiden dotierter Schichten Ionen in benachbarte Schichten diffundieren oder dotierte Schichten ihr Dotierungsprofil verändern, wenn der Wafer durch die Prozessierung zu stark erhitzt wird. Manchmal sind diese Effekte erwünscht oder eingeplant, häufig ist dies jedoch nicht der Fall.

Außerdem muss darauf geachtet werden, dass die aufgetragenen Materialien mit der Oberfläche kompatibel sind. Wenn darauf nicht geachtet wird, können unerwünschte Effekte auftreten. Spannungen im Material, das Entstehen Voltaischer Zellen oder gar das Abfallen von Teilen der aufgetragenen Schicht können die Folge sein.

Schichtveränderung

Neben den Verfahren, die neue Materialien auf den Wafer aufbringen, gibt es auch Verfahren, die bestehende Schichten ändern. Dabei ist die Unterscheidung zwischen einen schichterzeugenden und einem schichtverändernden Prozessschritt häufig schwer, da bei vielen Veränderungen auch eine neue Schicht entsteht.

Oxidation dient der Herstellung einer Siliziumoxid-Schicht aus einer bestehenden Silizium-Schicht. Das durch die Oxidation entstehende Oxid reicht zu ca. 40% in das Silizium hinein. Die verbleibenden 60% wachsen über die bestehende Oberfläche hinaus. Ist die Silizium-Schicht bereits teilweise mit anderen Materialien bedeckt, so kann dieses Material an den Randstellen angehoben werden. Dieser Effekt wird *birds beak* genannt. Oxid-Schichten können auch durch CVD Verfahren aufgetragen werden. Dann wird kein Silizium an der Oberfläche benötigt.

Dotierung verändert das Material, indem Fremdatome in das Material eingebaut werden. Die Dotierung verändert dabei nicht nur die Leitfähigkeit des Materials, sondern kann auch andere Eigenschaften verändern. So kann durch Dotierung auch die Ätzselektivität (siehe Ätzen) verändert werden. Die Topographie ändert sich bei der Dotierung im Normalfall nicht.

Auch hier ist zu beachten, dass der verändernde Prozessschritt nicht nur die Oberfläche betrifft, sondern auch tiefer liegende Schichten verändern

kann. Wird bei der Implantation von Ionen eine hohe Energiedosis verwendet, können die Ionen zum Beispiel in tiefere Schichten wandern.

Lithographie

Um Strukturen auf eine Schicht zu übertragen, wird in der Oberflächen-Mikromechanik eine Maske verwendet. Durch Beleuchtung mit einem UV-Laser wird die Struktur der Maske auf ein Polymer übertragen. Durch die Beleuchtung verändert sich die Struktur der Polymers. Je nach Typ können nach der Entwicklung beleuchtete oder unbeleuchtete Strukturen ausgelöst werden.

Neben der eigentlichen Lithographie gehören noch andere Prozessschritte zur Vor- und Nachbereitung der Strukturübertragung. So muss zuerst ein Polymer aufgetragen werden, das auf Bestrahlung reagiert. Dieses Polymer wird durch einen Prebake (Austreiben der Lösungsmittel) vor der Belichtung behandelt. Nach der Belichtung und Entwicklung findet oft noch ein Postbake (Härten des Polymers) statt.

Ätzen

Durch die Lithographie werden Strukturen lediglich auf ein Polymer übertragen. Meistens soll jedoch nicht das Polymer, sondern die darunterliegende Schicht strukturiert werden. Dies wird mittels Ätzverfahren realisiert. Bei den Ätzverfahren wird hauptsächlich zwischen Nass- und Trockenätzverfahren unterschieden.

Nassätzverfahren werden mit Hilfe von flüssigen Ätzmitteln mittels chemischer Wirkmechanismen durchgeührt. Der Vorteil dieser Verfahren ist die einfache Anwendung und die meist hohen Ätzraten. Durch Auswahl geeigneter Ätzmittel kann auch eine hohe Selektivität erreicht werden. Zur Durchführung wird im einfachsten Fall nur ein Gefäß mit der Ätzflüssigkeit benötigt. Nachteilig ist die geringe Kontrollierbarkeit der Ätzrate, die während des Ätzprozesses schwanken kann. Deshalb sind Maßnahmen, wie die Abscheidung von Ätzstopschichten, nötig, um den Ätzprozess zu kontrollieren.

Trockenätzverfahren arbeiten in der Regel mit gasförmigen Ätzmitteln. Da auch die Reaktionsprodukte meist gasförmig sind, ist der Abtransport gewährleistet. Der Ätzabtrag erfolgt chemisch durch reaktive Anteile im Ätzgas, physikalisch durch beschleunigte Ionen, oder durch eine Kombination von beiden. Rein physikalische Verfahren sind dabei meist nicht selektiv.

Bei der Auswahl des Ätzverfahrens achtet man vor allem auf *Selektivität* und *Isotropie*. Die Selektivität gibt das Verhalten des Ätzprozesses gegenüber verschiedenen Materialien an. Besitzt ein Ätzprozess eine hohe Selektivität, so wird das gewünschte Material sehr stark angegriffen, während die anderen Materialien nur einem schwachen Angriff ausgesetzt sind. Die Isotropie gibt die Richtungsabhängigkeit des Ätzverfahrens an. Ein isotroper Ätzprozess ätzt in alle Richtungen etwa gleich schnell. Bei anisotropen Verfahren gibt es eine Vorzugsrichtung. In Tabelle 2.2 werden Beispiele für verschieden Verfahren aufgeführt.

Die Geschwindigkeit des Ätzprozesses wird mit der Ätzrate gemessen. Die Ätzrate für einige spezielle Ätzprozesse wird in Tabelle 2.3 gezeigt. Dabei sind viele Umgebungsparameter, wie Druck und Temperatur, aber auch die verwendeten Anlagen von entscheidender Bedeutung. Deshalb können in verschiedenen Veröffentlichungen unterschiedliche Werte gefunden werden.

2.2.2 LIGA

Das LIGA Verfahren wurde in den 80er Jahren am damaligen Kernforschungszentrum in Karlsruhe (heute Forschungszentrum Karlsruhe) entwickelt. LIGA steht dabei für die verwendeten Fertigungsverfahren **LI**thographie, **G**alvanoformung und **A**bformung. Mit der LIGA Technik werden vor allem optische Bauelemente hergestellt. In Abbildung 2.2(a) ist ein Verbinder für Glasfaserkabel zu sehen. Dafür werden extrem präzise gefertigte Führungen benötigt, welche mit der LIGA Technik hergestellt werden. Aber auch mechanische Komponenten, wie Teile des Mikromotors, der den Fahrradfahrer in Abbildung 2.2(b) bewegt, werden mittels der LIGA Technik gefertigt. Beide Produkte wurden am IMM (Institut für Mikrotechnik, Mainz [Ins04a]) hergestellt.

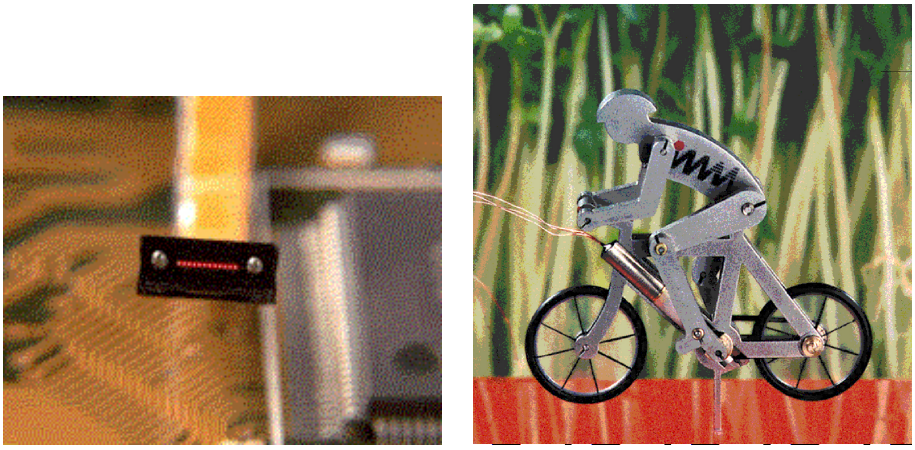
Ätz-Mechanismus	Anwendungen	typ. Ätzrate (bei Silizium)	Anmerkungen
nass-chemisch (isotrop)	isotrope Strukturen	50μm/min	schlecht kontrollierbar, einfach
nass-chemisch (anisotrop)	gewinkelte Strukturen, Brücken, Düsen	1μm/min auf (100)-Ebene	mit Ätzstopp kontrollierbar, einfach
trocken-chemisch	Entfernen von Resist, isotrope Strukturen	0.1-6μm/min	hohe Auflösung ($< 0.1\mu$m)
trocken-physikalisch	Reinigen der Si-Oberfläche, Entfernen dünner Schichten	300 Å/min	geringe Selektivität, Beschädigung der Kristallstruktur
trocken-physikalisch-chemisch	präzises Übertragen von Strukturen	0.1-6μm/min	wichtigstes Trockenätz-Verfahren

Tabelle 2.2: Verschiedene Ätz-Mechanismen im Vergleich [Sch04a].

Ätzprozess	$Si < 100 >$	SiO_2	Si_3N_4
$HF(49\%)$	-	23k	140
KOH	14k	77	0
$5:1BHF$	-	1000	9

Tabelle 2.3: Verschiedene Ätzprozesse im Vergleich, Angaben in Å/min.

Die LIGA Technik verwendet im Gegensatz zur Lithographie, die für die Oberflächen-Mikromechanik verwendet wird, keine UV-Laser als Strahlungsquelle. Als Quelle dient ein Teilchenbeschleuniger (Synchrotron). Dort werden Elektronen durch Magnetfelder beschleunigt, so dass eine Röntgenstrahlung ausgestrahlt wird. Diese Strahlung ist hoch parallel und linear polarisiert. Durch die aufwendigen Geräte (Teilchenbeschleuniger) und Spezialmasken, die für die Röntgenstrahlung erforderlichen sind, ist diese Art der Bestrahlung sehr teuer. Jedoch lassen sich mit der Röntgenstrahlung relativ dicke Polymerschichten



(a) Faserstecker (IMM)

(b) Fahrradfahrer im Kressefeld (IMM)

Abbildung 2.2: Mit LIGA hergestellte Produkte

sehr exakt strukturieren. Die erreichbaren Aspektverhältnisse liegen bei ungefähr 1:1000.

Das Polymer wird dabei auf einer Metallplatte aufgetragen. Nach der Belichtung und Entwicklung kann dann mittels Galvanik ein Negativ der Struktur des Polymers in Metall übertragen werden. Die so entstandene Metallform kann dann in Präge- oder Spritzgussverfahren auf beliebig viele Produkte aus Kunststoff abgeformt werden. Dadurch lassen sich die Kosten für die teure Technologie durch Massenfertigung amortisieren. Die abgeformten Kunststoffteile können wiederum als verlorene Formen für Keramiken oder andere Werkstoffe dienen. In Abbildung 2.3 ist der Ablauf des Verfahrens schematisch dargestellt.

2.2.3 Weitere Verfahren

Neben den vorgestellten Vorfahren gibt es noch viele weitere Möglichkeiten, um Strukturen im Mikrometer-Bereich herzustellen. Als erstes sind hier die Verfahren aus der Feinwerktechnik zu nennen. Hier können mit abgewandelten Bohr-, Fräs-, und Drehprozessen vor allem rein mechanische Strukturen hergestellt werden.

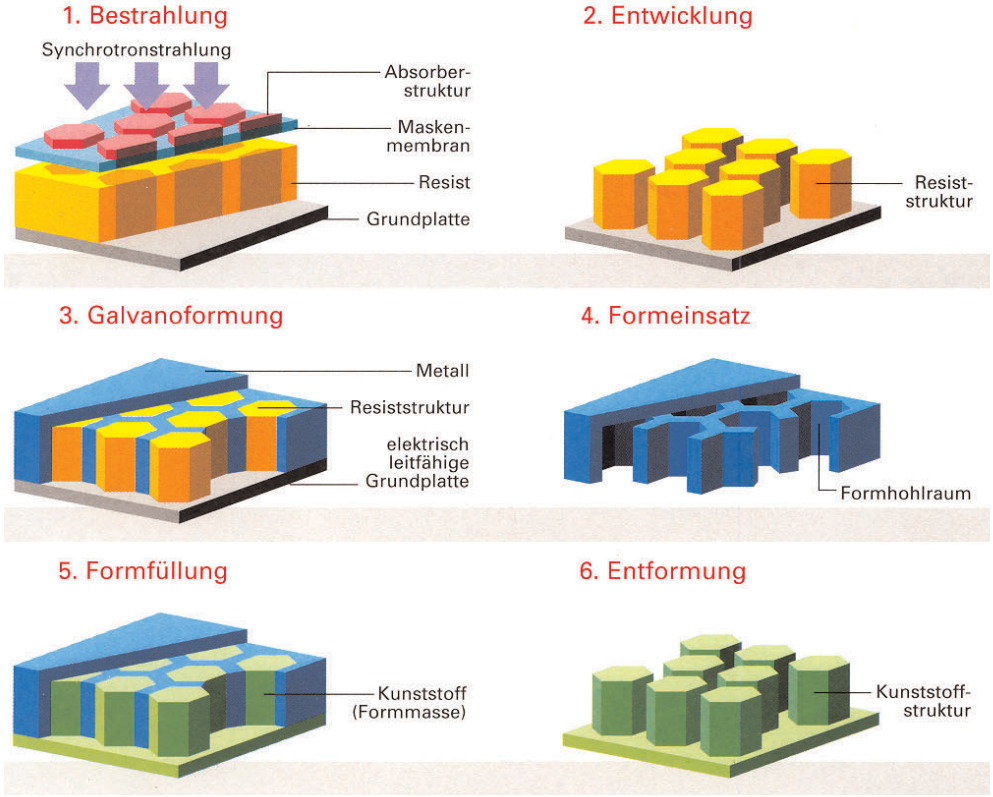


Abbildung 2.3: LIGA Verfahren [Ins04a]

Auch die Lasertechnik ist hier zu erwähnen. Sowohl in der Massenfertigung als auch im Rapid Prototyping finden Laser ihren Einsatz. So werden zum Beispiel die feinen Düsen eines Tintenstrahldruckers mittels Lasertechnik hergestellt. Andere Verfahren nutzen Laser, um direkt aus Polymeren 3-dimensionale Prototypen herzustellen.

Auch vom LIGA Verfahren gibt es Abwandlungen. Einer der Nachteile des LIGA-Verfahrens ist die teure Synchrotron-Strahlung. Deshalb gibt es eine Abwandlung, die mit UV-Belichtung arbeitet. Dieses Verfahren wird als Poor-Man's-LIGA bezeichnet.

Mit der Aufbereitung von Wissen über die Fertigung von nicht-siliziumbasierten Mikrotechniken beschäftigte sich das Transtec Projekt (siehe Kapitel 3.1.4 und [TP04, BRS01a]). Auf den Webseiten des Projektes sind genauere Informationen über das LIGA-Verfahren, die Laser Technik und die Feinwerktechnik zu finden.

2.3 Fazit

In diesem Kapitel wurden einige Materialien und Prozessschritte vorgestellt, die bei der Fertigung von Mikrosystemen Anwendung finden. Dabei wurde deutlich, dass jedes Material und jeder Prozessschritt durch eine Vielzahl von Parametern beeinflusst wird. So sind die Eigenschaften der Materialien zum Beispiel oft abhängig von der Art und Weise, wie sie erzeugt und nachbehandelt werden. Bereits eine geringe Änderung der Werte eines Parameters kann nachhaltige Auswirkungen auf das Ergebnis haben.

Auch die Ergebnisse der Prozessschritte können durch vorher oder nachher stattgefundene Prozessierung verändert werden. Vor allem Hochtemperaturschritte können tiefer liegende Schichten eines zu fertigenden Werkstückes stark verändern.

Alle diese Effekte lassen sich nur in den seltensten Fällen formal beschreiben. Deshalb ist es nötig, gesammelte Daten in einer organisierten Form zu speichern. Nur so können neue Erkenntnisse gewonnen werden. Im nächsten Kapitel werden einige bisherige Ansätze daraufhin untersucht, wie sie mit dieser Problematik umgehen.

3 Frühere und aktuelle Ansätze

Nachdem im vorherigem Kapitel deutlich wurde, dass in der Mikrosystemtechnik eine Vielzahl von Materialien und Prozessschritten verwendet werden, sollen in diesem Kapitel frühere und aktuelle Ansätze zum Umgang mit den anfallenden Daten untersucht werden. Im Mittelpunkt steht dabei die Untersuchung der verwendeten Datenhaltung und die Verwendbarkeit dieser in der Mikrosystemtechnik.

3.1 Universitäre Forschung

Der erste Abschnitt beschäftigt sich mit den im universitären Bereich entstandenen Systemen. Hierbei soll im Detail auf vier Systeme eingegangen werden, die dem Autor zu Testzwecken zur Verfügung standen.

3.1.1 SIMPL(er)

Das Tool SIMPL (SIMulated Profiles from the Layout) [Lee85] entstand in den 80er Jahren an der University of California, Berkeley. SIMPL ist ein CAD Tool, welches das Profil eines IC's bei gegebener Prozessfolge und vorhandenen Maskensatz erzeugt. Hauptziel der Software war es, dem Designer eine Visualisierung der durch sein Layout erzeugten Topographie zu ermöglichen.

Bereits 1985 konnten mit der Software Effekte wie „bird's beak", laterale Diffusion und die Abdeckung von nicht horizontalen Strukturen bei einer Deposition simuliert werden. Dabei wurden in der Version 2 Schnittstellen eingefügt, die es auch ermöglichten externe Simulatoren anzubinden.

Leider sind weder die Hardware noch das Betriebssystem, die für diese Software benötigt werden, heute noch in Betrieb. Ende der 90er Jahre wurde jedoch

eine vereinfachte Version der Software, passender Weise SIMPLer genannt, in Java portiert und steht frei zur Verfügung [Ele04].

Aufbau und Verwendung

Sowohl SIMPL als auch SIMPLer verarbeiten zwei Arten von Eingaben: Prozess- und Maskendaten. Die Prozessdaten wurden bei SIMPL über eine interaktive textuelle Schnittstelle eingegeben. Listing 3.1 zeigt die Eingabe der Daten für einen Abscheidungsprozess (Belackung). Die Eingaben für komplette Prozesse konnten in einer Datei zusammengestellt und im Batchbetrieb abgearbeitet werden.

```
WHICH PROCESS? DEPO
NAME OF THE MATERIAL? RST
THICKNESS OF THE MATERIAL (micro-meter)? 1.0
ISOTROPIC, ANISOTROPIC OR VERTICAL (I,A, or V)? I
DO YOU WANT TO DRAW THE CROSS SECTION (yes or no)? yes
```

Listing 3.1: Eingabe der Werte für eine Deposition in SIMPL

In SIMPLer steht zu diesem Zweck ein Editor mit Auswahlmenü zur Verfügung. Hier können entsprechende Prozessschritte ausgewählt und zur Prozessfolge hinzugefügt werden (siehe Abbildung 3.1).

Während in SIMPL die Layoutdaten aus einer Datei im CIF Format gelesen wurden, gibt es für die Maskendaten in SIMPLer einen einfachen grafischen Editor. In beiden Fällen wird ein repräsentativer Schnitt durch die Maske gemacht, um eine Strichmaske zu erhalten. Diese Maske wird dann verwendet, um Lithographieschritte zu simulieren.

Datenhaltung

Von einer echten Datenhaltung kann bei diesen Ansätzen noch nicht gesprochen werden. Beide Versionen (SIMPL und SIMPLer) bieten zwar die Möglichkeit entwickelte Prozessfolgen in einer Datei konsistent zu machen, aber die gespeicherten Daten beschränken sich auf ein Minimum. Eine echte Prozessfolge wäre aus diesen Dateien nicht wieder abzuleiten.

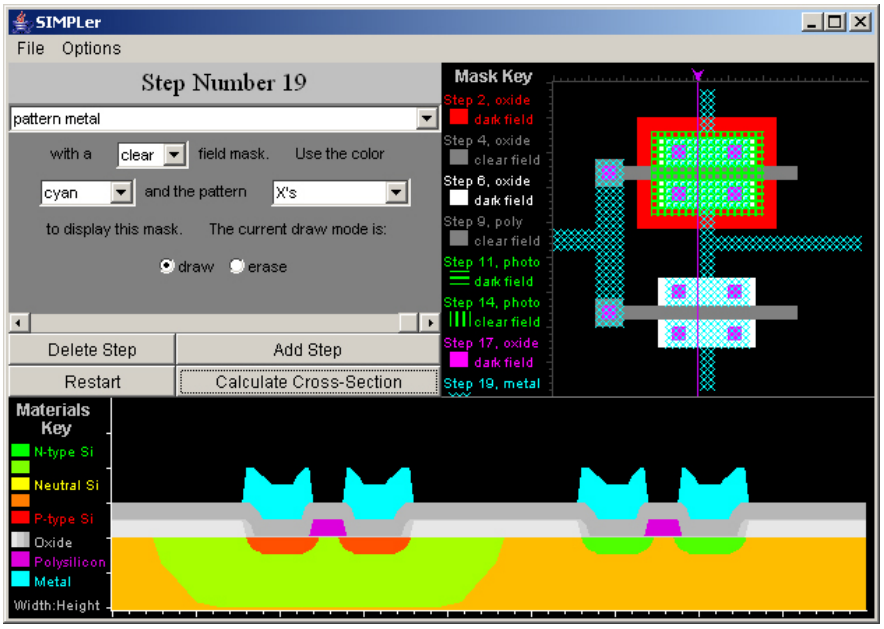


Abbildung 3.1: Erstellung von Prozessfolgen mit SIMPLer

Bewertung

Mit SIMPL wurde ein Tool entwickelt, das dem Maskendesigner helfen sollte, die Auswirkungen seiner Arbeit auf die Topologie des Entwurfs abschätzen zu können. Da es bereits zu Beginn der 80er Jahre entwickelt wurde, waren weder Hardware noch Software auf einem Stand, der heute allzuleicht für selbstverständlich angenommen wird. Deshalb sollen die hier erwähnten Kritikpunkte, wie auch bei den im folgenden vorgestellten Projekten, nur als „Ansporn“ für bessere zukünftige (und eigene) Entwicklungen stehen und nicht als Kritik an den Entwicklern!

Eines der ersten Probleme, die einem bei der Betrachtung von SIMPL auffallen, ist die geringe Konfigurierbarkeit. Alle Prozessschritte sind fest in die Software einprogrammiert und lassen sich nur durch Änderung des Quellcodes ändern. Dieser Ansatz macht die Software recht unflexibel.

Ein anderes Problem ist, dass die Software keinerlei Plausibilitätscheck vornimmt. Es wird angenommen, dass die eingegebene Prozessfolge korrekt ist. So können prinzipiell völlig unsinnige Prozessfolgen ohne Warnung simuliert werden. Dies mag bei den damals wenigen verfügbaren und festgeschriebe-

nen Prozessflüssen akzeptabel gewesen sein, heute ist es das ganz sicher nicht mehr.

Schließlich fällt auch auf, dass der Einfluss von Prozessschritten auf bestehende Materialien nicht berücksichtigt wird. In der Praxis ändern sich zum Beispiel Diffusionsprofile bei nachfolgenden Hochtemperaturschritten. Dies wird in der in [Lee85, Wu88] beschriebenen Version jedoch noch nicht berücksichtigt. Dieses Problem könnte jedoch durch die Anbindung externer Simulatoren gelöst werden.

3.1.2 MISTIC

Mit dem Aufkommen neuer Produkte aus den Bereichen Speicher (EERAM, EPROM, DRAM) und MEMS wurde es nötig, spezielle Prozessfolgen zu entwickeln. Die zur Verfügung stehenden Prozessfolgen (z. B. CMOS und Bipolar) reichten nicht mehr aus. Mit der Vielzahl neuer Prozessfolgen wuchs auch die Komplexität dieser. Die Verwendung der bis zu diesem Zeitpunkt angewandten Trial and Error Methoden wurde immer aufwendiger und zeitintensiver.

Mit dem Ziel eine Abhilfe für diese Problematik zu schaffen wurde MISTIC [ZCM99a, ZCM99b] entwickelt. MISTIC ist ein Akronym für Michigan Synthesis Tool for Integrated Circuits. Es wurde in den 90er Jahren an der University of Michigan entwickelt. Das Tool soll automatisch korrekte Prozessfolgen erzeugen. Der Fokus lag hierbei auf Dünnschicht Silizium Verfahren.

Aufbau und Verwendung

Das System besteht aus 4 Komponenten: dem Device Builder, dem Compiler, dem Process Viewer und dem Datenbankeditor. Alle diese Komponenten sind um eine zentrale Datenbank angeordnet. Der Program Manager (siehe Abbildung 3.2) steuert den Ablauf und den Datenfluss zwischen den Komponenten.

Die als Device Builder (Abbildung 3.3) bezeichnete Komponente ist ein grafischer Editor. Hier können Querschnitte von zu fertigenden Produkten erstellt werden. Gezeichnet werden können jedoch nur rechtwinklige Strukturen. Als erstes wird hierzu die Art des zu zeichnenden Blocks gewählt. Hierbei stehen

Abbildung 3.2: Die Benutzeroberfläche von MISTIC

Diffusion, Deposition, Reaktives Wachstum (reactive growth, z. B. Oxidation) und Opferschichtabscheidung (sacrificial type layers) zur Verfügung. Für jede dieser Typen können wiederum verschiedene Materialien gewählt werden. Nach Angabe der Dicke kann schließlich der Layer gezeichnet werden.

Der Compiler erhält als Eingabe den mit dem Device Builder erstellten Querschnitt. Danach wird die Struktur untersucht und in einen gerichteten Graph umgewandelt. Durch Operationen zum Zusammenfassen von Strukturen wird der Graph minimiert. Anschließend wird für die minimierte Struktur eine Prozessfolge aus Abscheidungs-, Diffusions- und Ätzschritten erstellt, die im letzten Schritt durch Lithographie und Reinigungsschritte ergänzt wird. Ein besonderes Augenmerk wurde auf die Erstellung von Diffusionsschritten gelegt. Die nötigen Energien und die Nachdiffusion durch spätere Hochtemperaturschritte werden berechnet, so dass das Ergebnis der kompletten Prozessfolge dem gezeichneten Polygon entspricht. Werden mehrere möglich Prozessfolgen gefunden, werden diese nach verschiedenen Kriterien (z. B. Robustheit des Prozesses, Kosten, etc.) bewertet ausgegeben.

Mit dem Process Viewer (Abbildung 3.4) können die vom Compiler erzeugten Prozessflüsse angezeigt werden. Hierzu stehen mehrere Möglichkeiten zur Verfügung. Als erstes kann man den Prozessfluss als Chargenkarte sehen und in verschiedenen Formaten ausgeben. Eine weitere Funktion ist die Generierung von Eingabedateien für TCAD Tools. Dies ermöglicht die Simulation des Prozesses und somit eine Überprüfung des Ergebnisses gegen den im Device

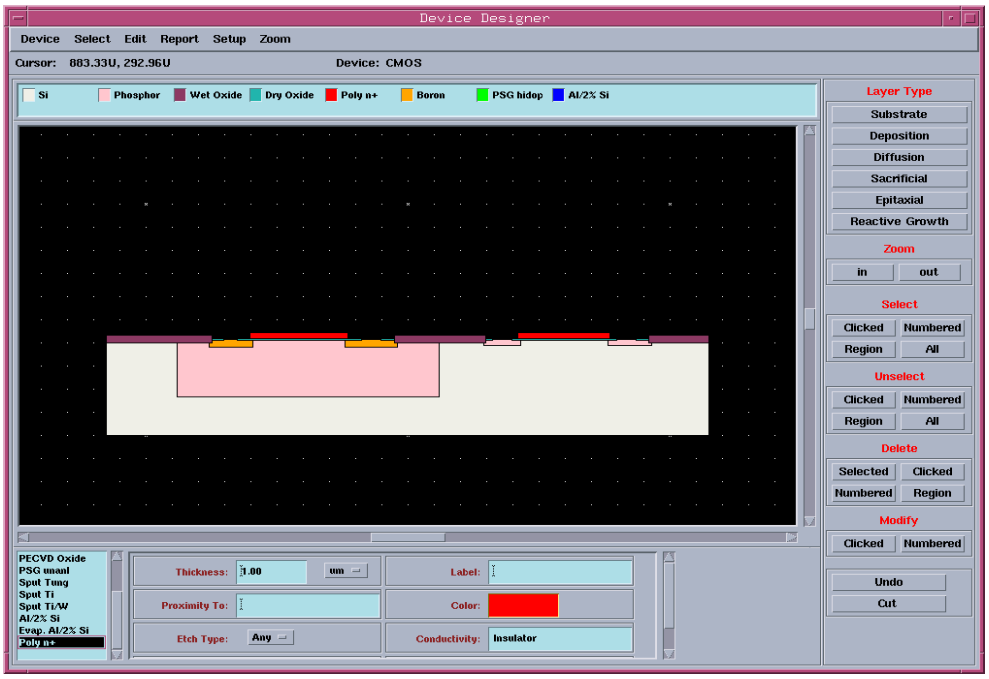


Abbildung 3.3: Device Builder (MISTIC)

Builder gezeichneten Entwurf. Schließlich kann auch noch ein Querschnitt angezeigt werden, in dem jeder einzelne Prozessschritt visualisiert werden kann. Mit diesem Teil des Viewers kann man eine Art Video erstellen, das die Prozessierung mit Effekten wie Nachdiffusion und Unterätzung darstellt.

Wie bereits anfangs erwähnt ist die Datenbasis das zentrale Objekt der Software. Sie enthält einzelne Module, die Atom genannt werden. Diese Atome enthalten Informationen über die einzelnen Prozesse. Der Datenbankeditor bietet die Möglichkeit für Deposition, Substrate, Diffusion (implant), Sacrificial (void), Epitaxial, Reactive growth, Photoresist, Etchant, Cleaning cycle und Lithography die Daten mittels einer grafischen Oberfläche zu Editieren.

Datenhaltung

Für die Datenhaltung in MISTIC wird das Dateisystem benutzt. Alle Daten sind in einer Datei namens `atom.dat` enthalten. Die Prozessschritte werden dabei als parametrisierte Rezepte gespeichert. Das Listing 3.2 zeigt als Beispiel einen Abscheideprozess. Eingeschlossen wird die Beschreibung in eine ATOM

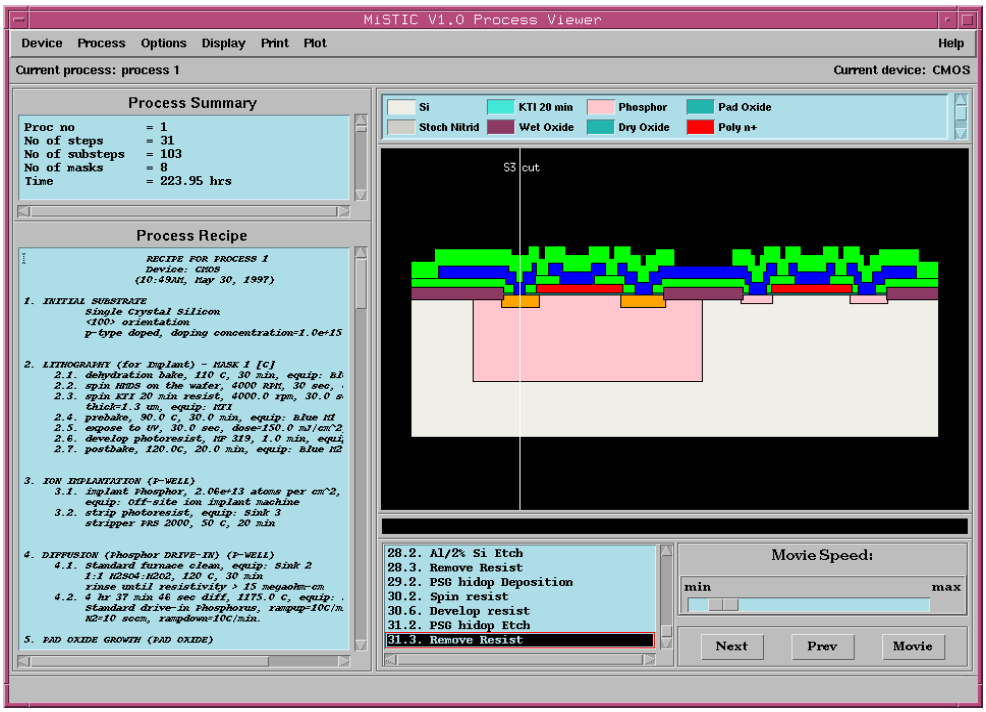


Abbildung 3.4: Process Viewer (MISTIC)

Umgebung (`\begin(ATOM)` und `\end(ATOM)`). Der `TYPE` gibt an, um welche Art von Prozessschritt es sich handelt. Im Beispiel steht das D für Deposition.

Der Punkt `CLEANING_TYPE` gibt einen Reinigungsprozess an. In diesem Fall handelt es sich um eine Piranha Reinigung. Dies ist ein in der Datenhaltung vorgesehene Mechanismus, um Prozessfolgen zu vervollständigen. Der angegebene Reinigungsprozess wird vom Compiler vor dem aktuellen Prozess eingefügt.

Danach folgen einige Parameter, die für die Simulation benötigt werden. Prinzipiell sind fast alle Prozessschritte lediglich durch die Zeit einstellbar, d.h. dass in der Beschreibung zeitabhängige Größen zu finden sind (z. B. `DEP_RATE` für Abscheiderate).

```
\begin{ATOM}
TYPE: D
GROUP_ID: 1
UNIQUE_NAME: lo_stress_nit
SHORT_NAME: Lo-stress Nit
SUPREM_NAME: NITRIDE
```

```
CLEANING_TYPE: piranha_clean
COLOR: gold2
CONDUCTIVITY: Insulator
DEP_RATE: 50
DEP_TEMP: 835
DEP_TEMP_MAX: 1050
DEP_SETUP_TIME: 3.5
DEP_FIXED_COST: 50
DEP_COST_RATE: 100
DEP_EQUIP: C4
DESCRIPTION:
.begin{description}
.end{description}
REFERENCE:
.begin{reference}
No Reference
.end{reference}
\end{ATOM}
```

Listing 3.2: Beschreibung eines Abscheideprozesses als Atom

Ätzprozesse können je nach angegriffenen Material unterschiedliche Ätzraten haben. In Listing 3.3 wird ein Beispiel für verschiedene Ätzraten gezeigt.

```

\begin{ATOM}
...
ETCH_RATE:
\begin{etch_rate}
lo_stress_nit 1100
dry_oxide 730
...
pad_oxide 730
\end{etch_rate}
...
\end{ATOM}
```

Listing 3.3: Beschreibung von unterschiedlichen Ätzraten

Auch für die Lithographie werden spezielle Konstrukte benötigt. Bevor ein Lithographieschritt durchgeführt wird, muss der Lack konditioniert werden (Prebake, Dehydration bake). Manchmal ist es auch nötig, einen Haftvermittler zwischen Untergrund und Lack aufzubringen. Dies wird durch die in Listing 3.4 gezeigten Konstrukte sichergestellt.

```
LITHOGRAPHY_ADHESION_PROMOTER: spin HMDS on the wafer,
        1333 RPM, 90 sec
LITHOGRAPHY_DEHYDRATION_BAKE: dehydration bake,
        110 C, 30 min
```

Listing 3.4: Prebake und Haftvermittler für Lithographie

Bewertung

MISTIC untersucht bei der automatischen Generierung von Prozessfolgen alle möglichen Varianten von Prozessschritten. Hierdurch erreicht man eine beinahe komplette Untersuchung des Lösungsraums. Dies hat jedoch auch zur Folge, dass die benötigte Rechenzeit exponentiell anwächst.

Ein weiteres Problem ist, dass die Prozessfolgen immer komplett generiert werden. Ist z. B. bereits ein Teilprozess bekannt, der nur erweitert werden soll, so ist diese Erweiterung mit MISTIC nicht möglich. Wünschenswert wäre die Möglichkeit zur Interaktion bzw. zur Vorgabe partieller Prozessfolgen.

Die Auswahl der Materialien für eine Schicht im Device Builder ist recht restriktiv. Der Nutzer muss bereits genau wissen, welches Material er mit welchem Prozessschritttyp erzeugen will. Vor allem in den frühen Phasen des Designs ist dies jedoch nicht immer klar. Das System sollte es ermöglichen, die Materialien freier zu spezifizieren (z. B. nach Leitwert, etc.).

Das MISTIC System setzt beim Entwurf auf das Prinzip correctness by construction. Alle durch das System erzeugten Prozessfolgen entsprechen den im System programmierten und durch das ATOM File parametrisierten Regeln. Deshalb findet auch keine Überprüfung der erzeugten Prozessfolge auf Plausibilität statt. Die im ATOM File anzugebenden Parameter beschränken sich jedoch auf Reinigungs-, Pre- und Postbake Schritte sowie auf Haftvermittler

für Lacke. Dies ist jedoch für komplexe Prozesse im MEMS Design problematisch, da neu entdeckte Regeln und Abhängigkeiten mit anderen Schritten nur durch Änderung des Quellcodes eingefügt werden können.

Wie bereits weiter oben erwähnt, werden alle Daten in einer Datei gespeichert. Die einzelnen Elemente (Atome) stellen Prozessschritte dar. Auch Materialien, wie z.B. Schicht- und Ätzmaterialien werden durch Prozesse (Deposition oder Ätzen) dargestellt. Dies hat den Nachteil, dass alle Daten mit Prozessschritten verbunden sind. So muss zum Beispiel beim Hinzufügen eines neuen Schichtmaterials jeder Ätzschritt angepasst werden.

Natürlich besteht auch hier das Problem, dass es mit zunehmender Zahl von Prozessschritten immer schwieriger wird, diese zu verwalten. Da keine Vererbungsmechanismen vorhanden sind, können Prozesse nur nach den Kategorien Deposition, Substrate, Diffusion (implant), Sacrificial (void), Epitaxial, Reactive growth, Photoresist, Etchant, Cleaning cycle und Lithography eingeordnet werden. Mit dem Datenbank Editor wurde zwar eine grafische Möglichkeit geschaffen, die Datei zu verändern, durch den Verzicht auf eine hierarchische Datenstruktur wird aber auch hier die Suche und das Editieren in großen Datenmengen erschwert.

3.1.3 LIDO

Der Name LIDO ist eine Kurzform von LIthographiebasierte Mikrotechnik und Dortmund. Es wurde Mitte der 90er Jahre an der Universität Dortmund entwickelt. Hauptzielsetzung war die Erstellung eines Prozesskonfigurations- und Regelprüfsystems, mit dessen Hilfe Prozessfolgen generiert und Layoutregeln für die Prüfung der Konsistenz von Designs in der Mikrosystemtechnik generiert werden konnten. Dabei wurden vor allem die Prozessschritte aus der LIGA - Technik (siehe Kapitel 2.2.2) zur Demonstration verwendet.

Aufbau und Verwendung

Das System besteht aus mehreren Komponenten, die der Nutzer im Rahmen des in Kapitel 4.2.2 beschriebenen Design-Flows durchläuft. In Abbildung 3.5

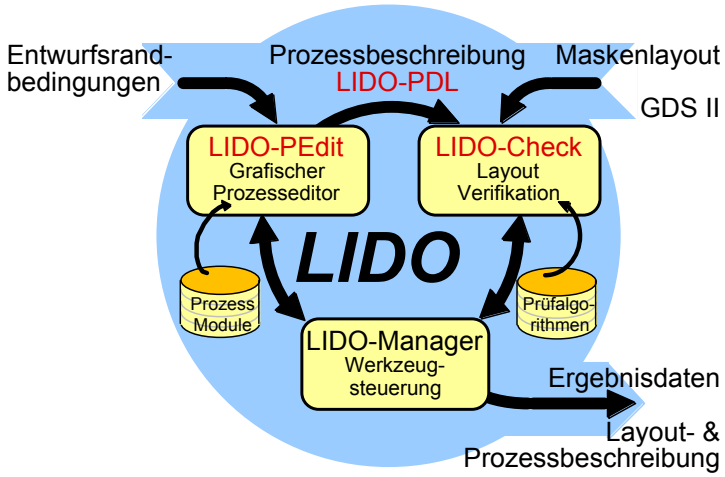


Abbildung 3.5: LIDO-Schema [Hah98a]

werden schematisch der Aufbau und die Komponenten des Systems dargestellt. Zunächst erfolgt der Entwurf des Layouts mittels eines konventionellen Layouteditors sowie die Festlegung einer Prozessfolge zur Herstellung des mikrotechnischen Produkts mittels des Editors LIDO-PEdit. Mit LIDO-Check kann schließlich das Layout auf Fehler geprüft werden.

Für die Erstellung einer Prozessfolge wurde der Grafische Prozessfluss Editor (GRAPE, später LIDO-PEdit) entwickelt. In diesem Editor kann ein Prozessfluss aus einzelnen Elementen zusammengestellt werden. Prinzipiell wird zwischen 3 Gruppen unterschieden:

Prozessschritte sind, wie der Name sagt, die eigentlich durchzuführenden Arbeitsschritte während der Herstellung eines Mikrosystems (z. B. Abscheidung, Ätzen, etc.).

Materialien werden in den Prozessschritten verwendet, zum Beispiel als Ätzmittel oder Dotierstoffe.

Masken werden in Lithographieschritten verwendet, um Strukturen zu übertragen.

Die einzelnen Objekte werden als Icons in drei den Kategorien entsprechenden Auswahlfenstern angezeigt. Mithilfe dieser Objekte kann mittels Drag &

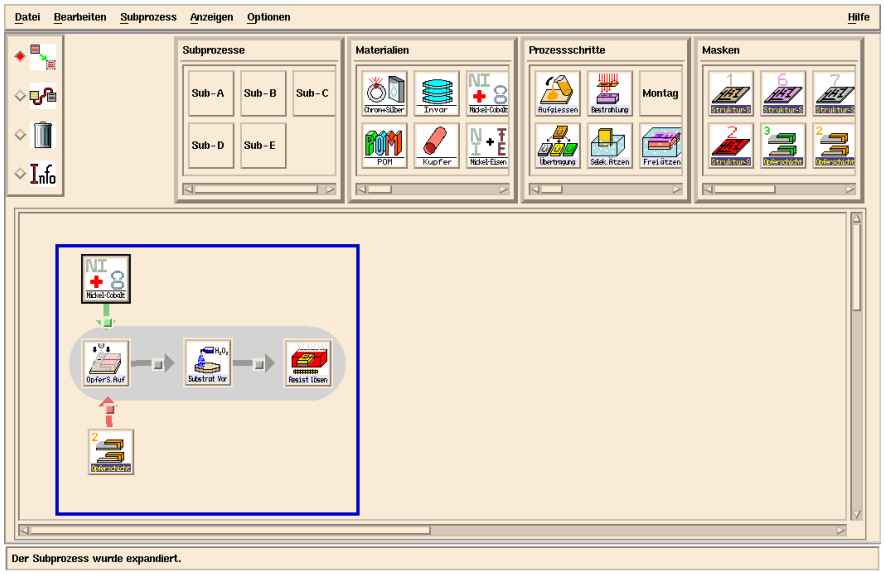


Abbildung 3.6: GRAPE: Grafischer Prozesseditor

Drop eine Prozessfolge erstellt werden (siehe Abbildung 3.6). Bei langen Prozessfolgen können Subprozessfolgen erstellt werden, die wiederum als Modul in der Hauptprozessfolge verwendet werden können.

Ein wichtiger Schritt bei der Erstellung ist die Konsistenzprüfung. Hierbei werden folgende Punkte geprüft:

Kompatibilitätsregeln: Legen die Verträglichkeit von Prozessschritten mit anderen Prozessschritten oder Materialien fest.

Reihenfolgenregeln: Legen fest, dass bestimmte Prozessschritte vor oder nach dem aktuellen Prozessschritt stattfinden müssen oder nicht stattfinden dürfen. Hierbei kann eingeschränkt werden, ob diese Vor-/ Nachbehandlung in unmittelbarer Nachbarschaft erfolgen muss.

Zyklenfreiheit: Bedeutet, dass Prozessfolgen keine Zyklen enthalten dürfen.

Materialzuordnung: Einigen Prozessschritt dürfen nur bestimmte oder gar keine Materialien zugeordnet werden

Layoutregeln: Die Layoutregeln in der Prozessfolge dürfen sich nicht widersprechen (z. B. width<50um und width>80um)

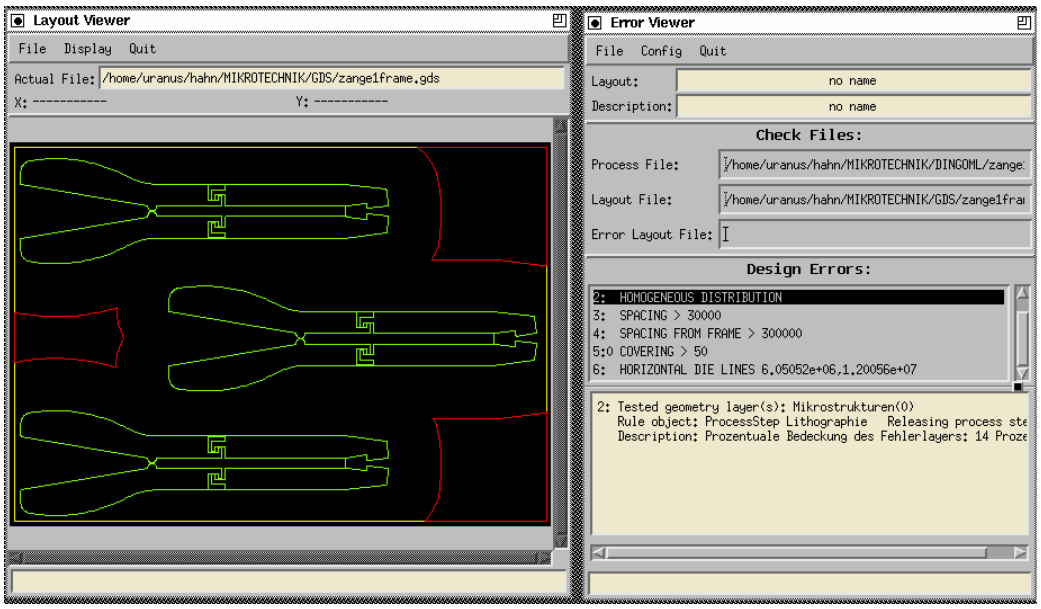


Abbildung 3.7: Fehlerausgabe bei LIDO-Check

Zur Prüfung werden zwei Mechanismen eingesetzt. Zum einen kann schon während der Eingabe eine Prüfung stattfinden, ob z. B. Materialien mit dem Prozess, dem sie zugeordnet werden sollen, kompatibel sind. Ein ausführlicher Konsistenzcheck kann am Ende der Prozessfolgenerstellung manuell gestartet werden. Das Ergebnis der Arbeit ist eine Prozessfolgenbeschreibung in der Sprache LIDO-PDL, welche weiter unten erläutert wird.

Ein weiteres Modul der Software ist der geometrische Entwurfsregelprüfer LIDO-Check. LIDO-Check erhält als Eingabe ein nicht hierarchisches Layout im GDSII Format und eine Beschreibung einer Prozessfolge in LIDO-PDL. Das GDSII Format wird in eine interne Datenrepräsentation (LIDO GeoDS) umgewandelt. Danach liest ein Parser die Beschreibung der Prozessfolge ein und durchsucht sie nach Layoutregeln und prüft die Geometrie auf die Einhaltung dieser. Werden Fehler festgestellt, so werden diese sowohl in textueller Form als auch im Maskview als Errorlayer dargestellt (siehe Abbildung 3.7).

Im Rahmen des LIDO Projekts wurde auch ein Optimierungsmodul vorgestellt. Die Sprache LIDO-PDL bietet die Möglichkeit, Verzweigungen in die Prozessfolge einzubauen. Anhand von durch den Nutzer spezifizierten Constraints kann der optimale Prozessfluss gefunden werden. Leider wurde dies nicht mehr in die grafische Oberfläche integriert.

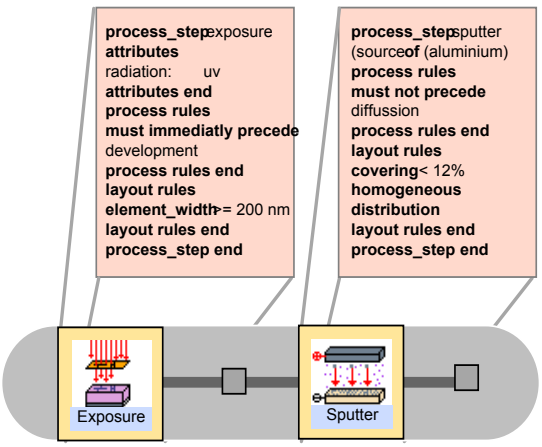


Abbildung 3.8: Zuweisung von PDL Code zu den entsprechenden Icons

Datenhaltung

Wie bereits mehrfach erwähnt, basiert LIDO auf der Sprache LIDO-PDL (Process Description Language). Eine genauere Beschreibung der einzelnen Objekte findet sich in [Hah98a, Hah98b]. Hier soll anhand einiger kurzer Beispiele die Verwendung der Sprache erläutert werden.

Um die Elemente der Prozessbeschreibung, die in GRAPE als Icons dargestellt werden, mit Leben zu füllen, werden sie mit Codefragmenten aus LIDO-PDL verbunden (siehe Abbildung 3.8). Das Listing 3.5 zeigt dies für einen Ätzprozess. Das Schlüsselwort `PROCESS_STEP` kennzeichnet den folgenden Quellcode als Beschreibung eines Prozessschrittes. In unserem Fall handelt es sich um ein Ätzverfahren, dass sowohl mit KOH als auch mit EDP durchgeführt werden kann. Als nächstes folgt ein Abschnitt `ATTRIBUTES`. Hier können allgemeine Informationen, wie Autor oder zu verwendendes Icon, aber auch für die Optimierung wichtige Informationen, wie z. B. Prozesskosten hinterlegt werden.

Die `PROCESS RULES` enthalten die Regeln zur Prozesskonfiguration. Prozessregeln bestimmen die Beziehungen von Prozessschritten untereinander (z. B. Reihenfolge, Vor- und Nachbehandlung) und die Verträglichkeit mit Materialien. Sie werden für den Konsistenzcheck verwendet. Die im Abschnitt `LAYOUT RULES` aufgeführten werden zuerst verwendet, um im Konsistenzcheck zu

prüfen, ob sich die Regeln widersprechen. Ist dies nicht der Fall, so werden diese Regeln zur Überprüfung des Layouts mit LIDO-Check verwendet.

```
PROCESS_STEP Aetzen (KOH, EDP)
   ATTRIBUTES
     Icon:"aetzen.xpm";
     Autor:"Jens Popp"
     Kosten:"200 EUR"
   ATTRIBUTES END
   PROCESS RULES
      MUST FOLLOW Fotolithografie;
      MUST IMMEDIATELY PRECEDE Reinigung;
   PROCESS RULES END
   LAYOUT RULES
      ANGLE >45;
      WIDTH >1 um;
   LAYOUT RULES END
PROCESS_STEP END
```

Listing 3.5: Beschreibung eines Ätzprozesses in LIDO-PDL

Das Listing 3.6 zeigt die Beschreibung eines Materials in LIDO-PDL. Der Aufbau ist ähnlich dem des Prozessschrittes. Natürlich benötigt man für Materialien keine Prozessregeln, aber das Layout kann durch Materialien beeinflusst werden. Auf ähnliche Weise lassen sich auch Masken- und Ressourcenobjekte (Maschinen) beschreiben (siehe [Hah98a, Hah98b]).

```
MATERIAL Aluminiumoxid
   ATTRIBUTES
      Farbe : "grau";
      Max_Temp : "200";
      Min_Temp : "160";
      Viskositaet : "";
      Durchlaessigkeit : "0.02  1 pro NM";
      Leitfaehigkeit : "0.5  Ohm pro cm";
      Haftung : "(PMMA,3),(Nickel,7)  Kraft pro cm^2";
      Min_Winkel : "30";
      Max_Winkel : "120";
```

```
      Icon : "alu_ox.xpm"
   ATTRIBUTES END
   LAYOUT RULES
      HEIGHT < 10 UM;
      ROUGHNESS = 20 NM;
      SHRINKING = 2 %;
      TOLERANCE = 5 %;
      WIDTH <= 20 UM;
      LENGTH >= 120 UM;
      AREA >= 200 UM^2
   LAYOUT RULES END
MATERIAL END
```

Listing 3.6: Beschreibung des Materials Aluminiumoxid in LIDO-PDL

Aus den beschriebenen Elementen lassen sich nun Prozessfolgen zusammenstellen. Eine einfache Prozessfolge bestehend aus den Schritten Oxidiation, Reinigung, Lithographie, Ätzen und Reinigung wird in Listing 3.7 gezeigt. Hierbei können die einzelnen Prozessschritte mit Materialien parametrisiert werden. Damit können Prozessschritte konkretisiert werden, die man mit verschiedenen Materialien oder Materialkombinationen verwendet. Somit werden Beziehungen zwischen Materialien und den Prozessschritten hergestellt, die sowohl für die Konsistenzprüfung (Materialunverträglichkeiten) als auch für die Geometrieprüfung (Layout Rules der Materialien) von Bedeutung sind.

Mit dem Maskenzuordnungsbezeichner `ON` kann man den Prozessschritten Masken zuweisen. Durch Klammerung sind im Beispiel den Prozessschritten *Lithographie* und *Ätzen* die Maske Oxid_Struktur zugewiesen. Hierdurch wird erreicht, dass alle Layoutregeln dieser Prozessschritte der Maske für den späteren Geometriecheck zugewiesen werden.

Der Maschinenzuweisungsbezeichner `WITH` ermöglicht es dem Prozessschritt ein RESSOURCE Objekt zuzuweisen. Hiermit können auch anlagenspezifische Parameter in spätere Berechnungen einfließen.

```
PROCESS drucksensor
    ATTRIBUTES
        NAME: "Beispielprozess";
```

```
    ATTRIBUTES END
    PROCESS_STEPS
        Wafervorbereitung;
        Siliziomoxidation ON Silizium_Layer WITH
            Oxidationsofen;
        Reinigung;
        (Lithographie (Positivlack) WITH Mask_Stepper;
         Aetzen (KOH) WITH Aetzanlage) ON Oxid_Struktur;
        Reinigung
    PROCESS_STEPS END
PROCESS END
```

Listing 3.7: Prozessfluss in LIDO-PDL

Innerhalb eines `PROCESS` Konstruktes können auch alternative parallele Prozessflüsse aufgebaut werden. Hierzu werden die Schlüsselworte `(ALT)SPLIT, (ALT)JOIN` und `(ALT)PAR` verwendet. Eine nähere Beschreibung dieser Konstrukte findet sich in [Hah98a].

Vererbung

Einige der Prozesselemente, wie Prozessschritte oder Materialien unterscheiden sich oft nur marginal voneinander. Dies ermöglicht es, Elementklassen zu bilden, in denen die gemeinsamen Eigenschaften gespeichert werden. Die konkreten Materialien und Prozessschritte können dann als eine Art Ableitung von diesen Klassen angesehen werden.

Die Sprache LIDO-PDL verwendet bereits einfache objektorientierte Konzepte. Der Vererbungsmechanismus wird durch das Schlüsselwort `IS` gekennzeichnet. Im Listing 3.8 wird ein Beispiel für diesen Mechanismus gezeigt.

```
MATERIAL polymer
    LAYOUT RULES
        common rules
    LAYOUT RULES END
MATERIAL END
```

```
MATERIAL pmma IS polymer
MATERIAL END

MATERIAL ps IS polymer
    LAYOUT RULES
        additional rules
    LAYOUT RULES END
MATERIAL END
```

Listing 3.8: Beispiel für Vererbung in LIDO-PDL

Bewertung

Wie bei den meisten universitären Projekten handelt es sich bei LIDO nur um eine prototypenhafte Implementierung. Sowohl die Sprache LIDO-PDL als auch die Software sind in mehreren Schritten erweitert und angepasst worden, so dass sich teilweise Inkonsistenzen zwischen der Sprachbeschreibung und den Auswertealgorithmen eingeschlichen haben. Leider sind auch einige der beschriebenen Funktionalitäten (Optimierung, Vererbung) nicht in der grafischen Oberfläche umgesetzt.

Mit der Vererbung wurde ein interessanter Mechanismus eingeführt, um redundante Wiederholungen und somit Fehler zu vermeiden. Dieses Konzept ist jedoch nur in der Sprachdefinition und nicht in der Software umgesetzt. Außerdem fehlt eine Festlegung, wie das Überschreiben von Attributen und Regeln umgesetzt werden soll (was passiert, wenn in der Klasse ein Intervall für `WIDTH` festgelegt wird, in der konkreten Umsetzung aber ein Wert außerhalb dieses Intervalls gewählt wird?).

Mit den Prozess- und Layoutregeln wurde ein Mechanismus geschaffen, um die Konsistenz von Prozessfolgen zu prüfen. Die vorgestellten Regeln sind jedoch nicht ausreichend. Bedingte Prozessregeln sind noch nicht vorgesehen, werden in der Praxis jedoch häufig benötigt (z. B. HMDS wird vor dem Auftragen des Lacks abgeschieden, wenn der Untergrund ein Silizium ist.)

Die Darstellung von Prozessen mit Icons hat sich in der Praxis als nicht praktikabel herausgestellt. Bereits einfache Prozessfolgen besitzen mehr als 20 Prozessschritte. Somit wird eine Darstellung als Graph äußerst unübersichtlich.

Das größte Problem ist jedoch, dass es keine Unterstützung für die Erstellung von Objekten für den Prozessflusseditor gibt. Die Materialien und Prozessschritte sind lediglich mit Hilfe eines Texteditors erstellbar und werden als Dateien mit der Endung .ml in einem dafür vorgesehenen Verzeichnis abgelegt. Beim Programmstart werden alle Dateien eingelesen und in Objekte der grafischen Nutzeroberfläche umgewandelt. Dies hat mehrere Nachteile:

- Die Eingabe der Daten von Hand ist fehlerträchtig, da Tippfehler nicht sofort erkannt werden.
- Es können unter Umständen sehr viele Materialien und Prozessschritte entstehen. Die Verwaltung vieler einzelner Dateien im Dateisystem wird dann schnell unübersichtlich.
- Die Software muss alle Dateien einlesen. Dies kann bei wachsender Anzahl den Speicherplatz sehr belasten.

3.1.4 Transtec und Interlido

Das Projekt Transtec [Pri04, TP04] beschäftigt sich mit der Vermittlung von Wissen im Bereich der Mikrosystemtechnik. Auch wenn dies nicht direkt mit den bisher beschriebenen Tools in Verbindung zu bringen ist, so werden doch einige interessante und neue Aspekte beleuchtet.

Die Motivation für dieses Projekt bestand in der Problematik, dass Ingenieure häufig nicht die Zeit haben, sich in ihrem Fachgebiet weiter zu bilden. Der Besuch von Seminaren und Konferenzen ist für die meisten kleinen und mittelständischen Unternehmen (KMU), wie sie in der Mikrosystemtechnik häufig anzutreffen sind, zu kostspielig.

Aufbau und Verwendung

Das Transtec System basiert auf einem Content Management System der Firma Hyperwave [Hyp04]. Dieses System wurde für die Anwendung in E-Learning

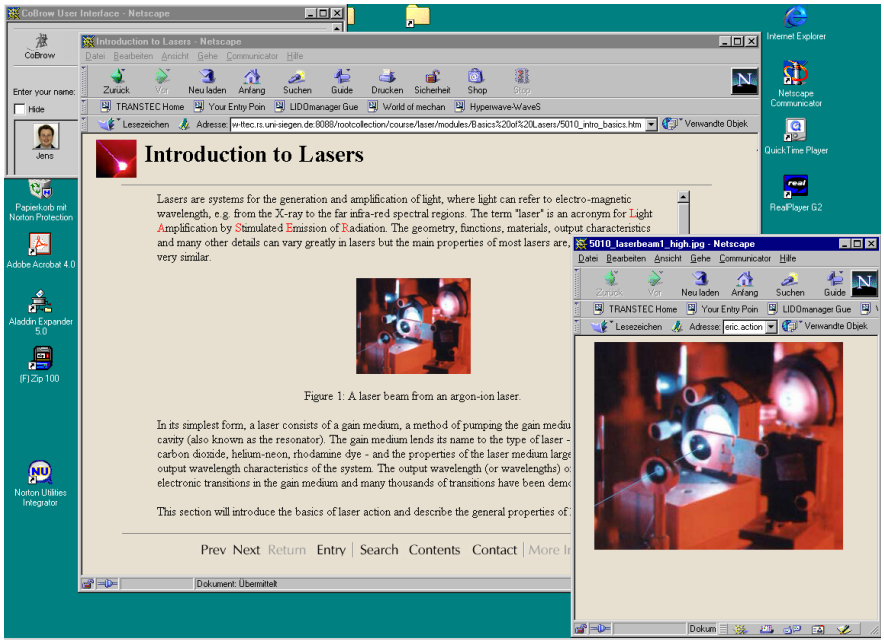


Abbildung 3.9: Arbeitssitzung mit dem Transtec Server

Systemen entwickelt. Durch so genannte Templates ist die Möglichkeit gegeben, Inhalte mit einem einheitlichen Layout zu versehen. Darüber hinaus können durch die Templates bestimmte Funktionalitäten, wie z. B. eine dynamische Navigation, erzeugt werden. Abbildung 3.9 zeigt eine typische Arbeitssitzung mit Transtec.

Zusammen mit mehreren Partnern aus Industrie und Forschung wurden im Rahmen dieses Projektes neben den Templates auch Inhalte erzeugt. Mehrere Kurse aus der Mikrosystemtechnik zu den Themen LIGA, Lasertechnik und Feinwerktechnik stehen zur Verfügung.

Die entstandenen Inhalte lassen sich bequem mittels eines Webbrowsers anschauen. Dies hat den Vorteil, dass keine spezielle Hard- oder Software benötigt wird und der Zugriff von jedem Punkt auf der Erde mit entsprechendem Netzanschluss erfolgen kann. Besonders Ingenieure in den KMU's aber auch interessierte Studenten können sich auf diese Weise fortbilden.

Der Transtec Server ist in mehrere Schichten unterteilt. Die Daten (Inhalte der Kurse, Bilder, Videos, etc.) werden in einem Datenbankmanagementsystem verwaltet. Ein WaveMaster genannter Server bereitet die Daten auf und leitet

sie an den Webserver weiter. Schließlich kann der Webbrowser des Anwenders als Client angesehen werden.

Im Rahmen des Transtec-Projektes wurde auch die Verwendung von Internet-basierten Werkzeugen für die Mikrosystemtechnik untersucht. Motivation für diese Untersuchungen war die Erkenntnis, dass die Softwarewerkzeuge in diesem Bereich sehr teuer und schwer zu handhaben sind und deshalb in den KMU nur sehr beschränkt zum Einsatz kommen. Mit dem Internet eröffnet sich die Möglichkeit, diese Tools kostengünstig in einem „Pay per Use“ Verfahren zur Verfügung zu stellen. Als proof of concept wurde INTERLIDO, eine Implementierung von Teilen der LIDO Software in Java, online gestellt.

Datenhaltung

Für die Verwaltung der Dokumente wird ein Content Management System auf Basis eines Datenbankmanagementsystems eingesetzt. Dies erlaubt durch die Erteilung von Rechten einen sehr feingranularen Zugriffsschutz. Jedem Dokument können so genannte Metadaten zugewiesen werden. Metadaten sind „Daten über Daten“. So kann man zum Beispiel Bild und Tondokumenten eine textuelle Beschreibung zuweisen, die ein späteres Wiederauffinden der Daten durch eine Suchmaschine ermöglichen. Die Metadaten werden jedoch auch genutzt, um die Navigation durch Kurse zu erleichtern und Querverweise auf nützliche Texte zu ermöglichen.

Interlido basiert wie bereits erwähnt auf LIDO. Deshalb wurden auch hier Dateien verwendet, um Elemente wie Materialien, Prozessschritte und Masken in LIDO-PDL zu beschreiben.

Bewertung

Mit Transtec wurde erstmals die Vermittlung von Wissen über die Mikrosystemtechnik mit der Verwendung von Tools verbunden. Studenten und Ingenieuren sollte die Möglichkeit gegeben werden, ihr Wissen zu erweitern und sich mit aktuellen Werkzeugen vertraut zu machen. Obwohl die entstandenen Kurse als State of the Art gelten können und auch als Buch erschienen

sind [BRS01b], so leiden sie doch unter der Tatsache, dass sie seit Ende des Projektes nicht mehr gepflegt wurden. Ein solches System ist nur dann sinnvoll, wenn die Informationen regelmäßig auf den aktuellen Stand gebracht werden.

Leider ist auch die Umsetzung von Interlido nur als Proof of concept anzusehen. So ist es z. B. nicht möglich eigene Prozessschritte zu definieren. Trotzdem sind die mit Transtec und Interlido gemachten Erfahrungen wertvoll. So wurde zum Beispiel im Rahmen dieses Projektes zum ersten mal der Einsatz von verteilten Systemarchitekturen (siehe auch Kapitel 7) für Werkzeuge in der Mikrosystemtechnik getestet. Ein weiteres Ergebnis dieser Arbeiten ist, dass das Internet die Möglichkeit bietet, sowohl das Hintergrundwissen als auch die benötigte Software kostengünstig zu verbreiten. Auch wenn aus Datenschutzgründen sicherlich keine kommerziellen Entwürfe über das Internet durchgeführt werden, so wird doch die Hemmschwelle zur Nutzung neuer Tools erheblich gesenkt. Außerdem erleichtert diese Art der Verbreitung kleinen und mittelständischen Unternehmen (KMU) den Test neuer Entwurfs-Software ohne hohe Anfangskosten für neue Hard- und Software.

3.1.5 Sonstige Forschungsprojekte

Neben den im Detail vorgestellten und abgeschlossenen Projekten, laufen zur Zeit auch einige andere interessante Arbeiten zu diesem Thema. Leider sind die Programme nicht zum Test verfügbar, so dass keine qualifizierte Aussage über diese Programme gemacht werden kann. Nichtsdestotrotz sollen hier zwei weitere Systeme erwähnt werden, um zu zeigen, dass die in späteren Kapiteln erwähnten Konzepte auch von anderen Gruppen angewandt werden.

In Braunschweig wird am Institut für Mikrotechnik im Rahmen des Sonderforschungsbereiches 516 „Konstruktion und Fertigung aktiver Mikrosysteme“ [WG04] das System Rumtopf entwickelt. Ähnlich wie LIDO wird Rumtopf zur Überprüfung von Prozessfolgen eingesetzt. Hierbei werden neben den Prozesskonsistenzen auch Materialeigenschaften überprüft. Interessant ist hierbei, dass ein Datenbanksystem zur Speicherung der gesammelten Daten und Regeln verwendet werden soll [HB01, HTB+03, HGBF03].

In [ZD02, ZD03] wird ein webbasiertes System zur Unterstützung des Entwurfs in der Mikrosystemtechnik vorgestellt. Das System setzt auf eine große

Wissensbasis, um den Entscheidungsprozess für Materialien und Prozessschritte zu unterstützen. Auch hier wird ein Datenbanksystem zur Verwaltung der Daten vorgeschlagen. Interessant ist auch die Verwendung einer Mehrschicht Systemarchitektur auf Basis von Java. Dies ermöglicht eine Zusammenarbeit von Experten auf verschiedenen Gebieten.

3.2 Industriell angewandte Systeme

In diesem Abschnitt sollen einige in der Industrie eingesetzte Systeme untersucht werden. Für die durch Softwarefirmen hergestellten Systeme sind jedoch nur in sehr begrenztem Maße Informationen über die Art der Datenhaltung verfügbar. Deshalb kann die Betrachtung dieser Systeme nicht in derselben Tiefe durchgeführt werden, wie dies für universitäre Systeme geschehen ist.

3.2.1 IP Management

IP steht im Zusammenhang mit der Entwicklung von elektronischen Schaltungen nicht für Internet Protokoll sondern für *Intellectual Property*. Die Idee der IP ist es, bereits fertig entwickelte Schaltungsmodule anderen Nutzern - meist gegen Bezahlung - zur Verfügung zu stellen. Ein IP kann dabei vom einfachen Gatter bis hin zur fertigen CPU alles enthalten.

Prinzipiell unterscheidet man nach [Con99] zwischen:

Soft cores sind meist unabhängig von der verwendeten Technologie. Sie können skalierbar (z. B. 32 oder 64 Bit Busbreite) sein und sind in einer Hochsprache wie VHDL oder Verilog beschrieben. Die Abmessungen und die realen Laufzeiten sind unbekannt und werden nur als Abschätzung angegeben. Sie werden erst durch die Implementierung in einer Zieltechnologie bestimmt.

Hard cores sind abhängig von der verwendeten Technologie. Sie sind nicht skalierbar und optimiert auf Performance, minimale Größe oder niedrigen Energieverbrauch. Für die Simulation sind Laufzeiten und Abmessungen bekannt.

Firm cores sind eine Mischung aus Hard- und Soft cores. Sie sind bedingt auf andere Technologien portierbar. Jedoch sind mehr Informationen über die Leistung der Schaltungen verfügbar.

Das Management von IP's basiert heute meist nur auf dem Austausch von Dateien, die die entsprechenden Daten enthalten. Eine geordnete Verwaltung gibt es - wenn überhaupt - nur bei einzelnen Herstellern, nicht aber herstellerübergreifend. Auch haben viele Hersteller besonders für verschlüsselte IP-Cores eigene Standards entwickelt. Mit Open Access soll ein Standard eingeführt werden, der die Vielfalt der verwendeten Datenformate begrenzt und herstellerübergreifend Daten wie IP's zur Verfügung stellt.

3.2.2 Open Access

In den letzten Jahren haben sich auf dem Markt mehrere Hersteller von EDA[1] Software etabliert und auf verschiedenen Gebieten spezialisiert. Deshalb enthalten heute Tool-Flows einiger Hersteller von integrierten Schaltkreisen EDA-Software verschiedener Hersteller. Dies wirft das Problem auf, die Ausgaben des einen Tools als Eingabe für ein anderes Tool verwenden zu können. Bisher werden hierzu die Dateiformate der einzelnen Hersteller ineinander konvertiert, was einen großen Mehraufwand an Zeit und Ressourcen bedeutet.

Open Access [Sil04a, Bal03] ist eine Initiative führender Hersteller von EDA Software und von Anwendern dieser Software. An dem Konsortium sind unter anderem Cadence Design Systems, Mentor Graphics und STM beteiligt. Ziel ist es, eine Schnittstelle für das Design von integrierten Schaltungen zur Verfügung zu stellen, die die Interaktion der einzelnen Tools ermöglicht. So soll zum Beispiel ein Synthese-Tool von Mentor mit dem Layout Tool von Cadence zusammenarbeiten können, ohne dass die umständliche Konvertierung der Dateien durchgeführt werden muss.

Aufbau und Verwendung

Open Access an sich stellt nur die Spezifizierung der Schnittstelle zur Verfügung. Zwar gibt es eine Referenzimplementierung der Datenbank, diese kann

[1] Electronic Design Automation

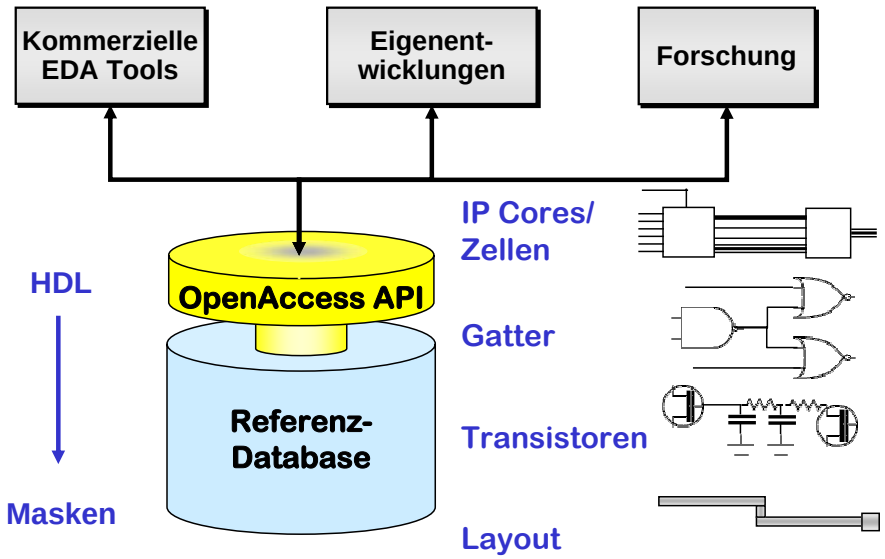


Abbildung 3.10: Aufbau von Open Access [Das03]

jedoch von den Herstellern selbst neu implementiert werden. Die Datenbank beinhaltet dabei alle Designstufen von komplexen IP-Cores bis hin zum Polygon des Layouts. Die Schnittstelle beinhaltet Konstrukte, die die nötige Aussagekraft besitzen, um alle Design Werkzeuge von der Register Transfer Ebene bis zur Generierung von GDSII Dateien (siehe Abschnitt 3.3.1) zu unterstützen.

Abbildung 3.10 zeigt den prinzipiellen Aufbau einer Architektur, die Open Access verwendet. Über der Datenbankschicht liegt eine API (Application Programmers Interface), die die Schnittstelle bereit stellt. Über diese API können die Programme unterschiedlicher Hersteller zusammen mit Eigenentwicklungen auf eine gemeinsame Datenbasis zugreifen.

Bewertung

Open Access stellt sicherlich eine der bedeutendsten Neuerungen in der EDA Industrie in den letzten Jahren dar. Leider sind jedoch die Spezifikationen noch nicht fertig gestellt. Besonders die im Rahmen dieser Arbeit interessanten Aspekte des fertigungsnahen Entwurfes sind noch nicht behandelt. Wenn für diese Aspekte eine Spezifikation fertig gestellt ist, wird auch eine Anbindung an Open Access notwendig.

3.2.3 Silvaco

Die Simulation von Prozessfolgen zur Fertigung eines mikroelektronischen Produktes gewinnt auch in der IC-Industrie an Bedeutung. Bei immer kleiner werdenden Strukturen beeinflusst die Technologie zur Fertigung immer stärker die Funktion. Kenntnisse über die Struktur der Schichtfolge und die Verteilung der Dotierung ermöglichen zum Beispiel Vorhersagen über das Verhalten eines Transistors, bevor dieser gefertigt wurde.

Silvaco (**SIL**icon **VA**lley **CO**mpany) [Sil04b] stellt unter anderem Software zur Simulation solcher Prozessfolgen her. Das Hauptaugenmerk liegt dabei (bis jetzt) auf der Simulation von mikroelektronischen Strukturen. Die Ergebnisse der Simulation werden genutzt, um Spice-Modelle zu extrahieren. Diese können von anderen Software-Tools zur Simulation auf einer höheren Ebene genutzt werden. So kann bereits vor der Fertigung einer Schaltung eine Abschätzung über deren Verhalten gegeben werden. Durch die Verwendung von Modellen auf einer abstrakteren Ebene lassen sich auch komplexere Schaltungen modellieren.

Da viele Prozessschritte aus der Mikroelektronik auch in der siliziumbasierten Mikrosystemtechnik verwendet werden, bietet es sich natürlich an, die Simulatoren auch hierfür zu verwenden. Deshalb ist Silvaco auch ein Partner im EU Projekt Promenade. Die Palette der von Silvaco angebotenen Software ist recht umfangreich. Deshalb soll hier nur der eigentliche Simulator und die Werkzeuge zum Start und zur Anzeige kurz vorgestellt werden.

Aufbau und Verwendung

Die Simulationssoftware besteht aus mehreren Komponenten. Der Nutzer startet die Simulation aus dem Programm `deckbuild`. Hier können einzelne Befehle eingegeben oder ein deck-file geladen werden. Abbildung 3.11 zeigt die Software mit einer geladenen Simulations-Anweisung.

Wenn alle Befehle zur Simulation eingegeben beziehungsweise geladen wurden, kann der eigentliche Simulator Athena gestartet werden. Die Software geht bei der Simulation davon aus, dass die eingegebenen Daten korrekt sind. Eine Plausibilitätsprüfung findet nicht statt. Als Ergebnis entsteht ein

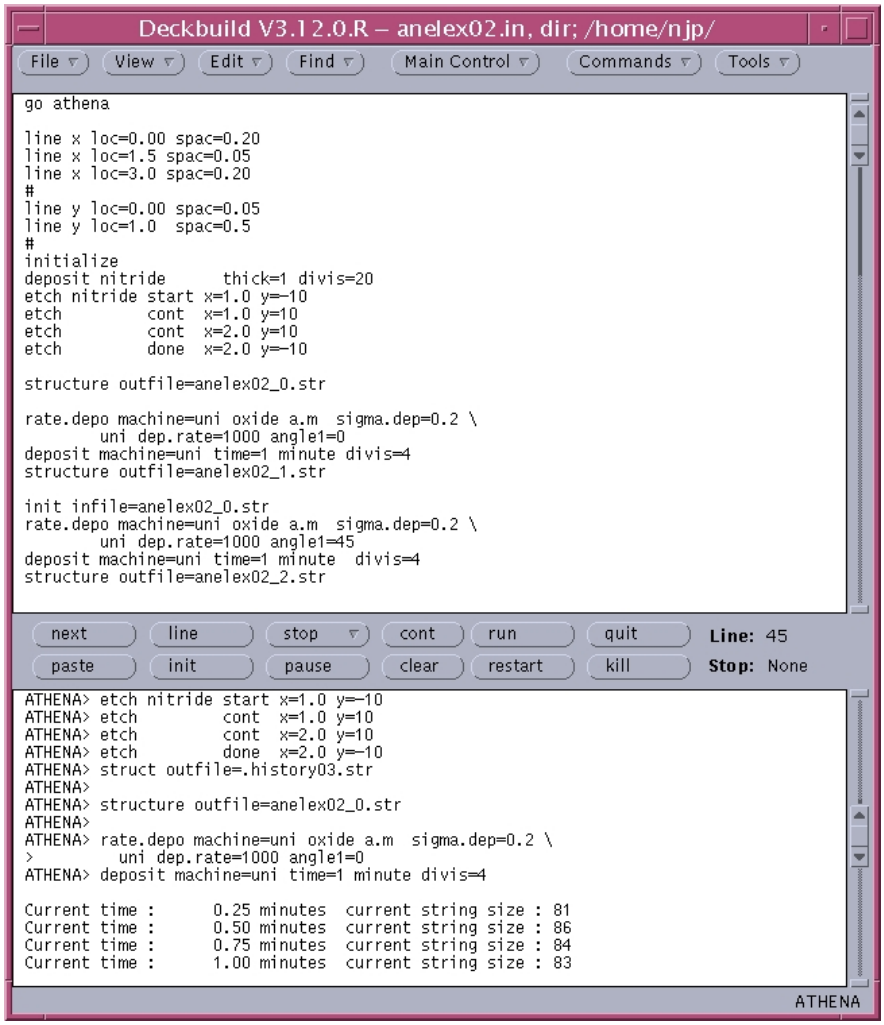


Abbildung 3.11: Deckbuild

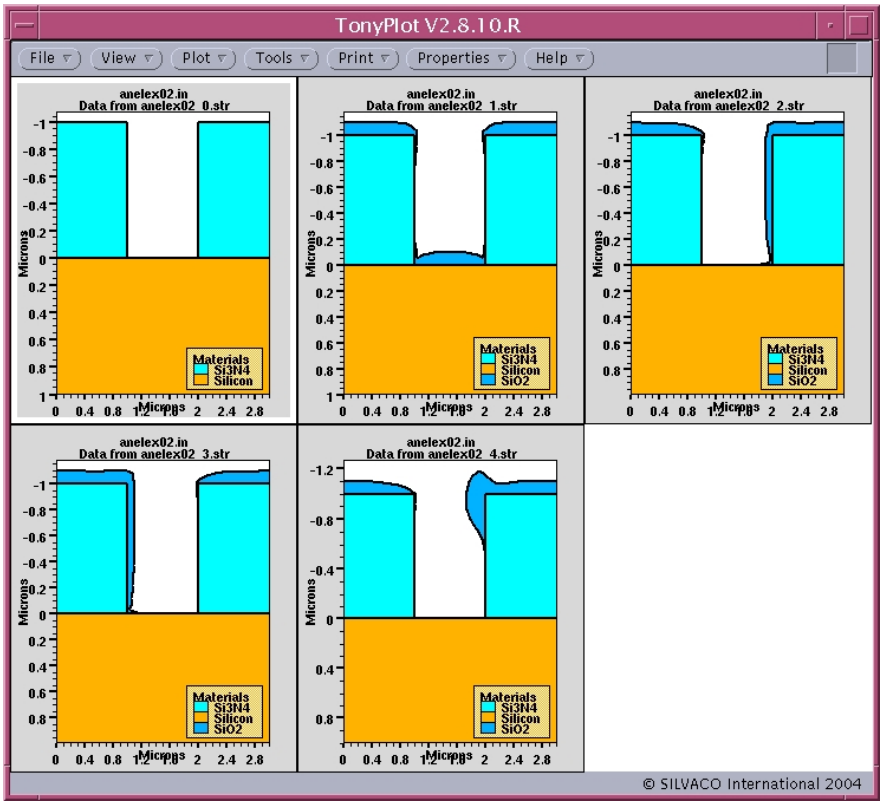


Abbildung 3.12: TonyPlot

Structure-file, dass mit dem Anzeigeprogramm TonyPlot visualisiert werden kann. Je nach Simulation können hier auch die Dotierungsdichte oder ähnliche Effekte sichtbar gemacht werden. Abbildung 3.12 zeigt das Ergebnis der Simulation einer Abscheidung, wobei der Wafer in unterschiedlichen Winkeln zur Abscheidungsquelle stand.

Datenhaltung

Die Datenhaltung unterteilt sich in zwei Segmente. Zum einen werden in der Software Modelle zur Simulation verwaltet. Diese Modelle sind in Form von Programm-Modulen in die Software eingebunden oder können über einen Interpreter als Quellcode zugeführt werden.

Das zweite Segment sind die Nutzerdaten. Diese werden in Form von sogenannten Input-Decks in Dateien gespeichert. Ein Input-Deck kann hierbei wiederum andere Input-Decks nachladen, so dass eine gewisse Hierarchie aufge-

baut werden kann. Listing 3.9 zeigt Abschnitte des Codes für das in Abbildung 3.12 gezeigte Beispiel.

```
go athena

line x loc=0.00 spac=0.20
line x loc=1.5 spac=0.05
line x loc=3.0 spac=0.20
#
line y loc=0.00 spac=0.05
line y loc=1.0  spac=0.5
#
initialize
deposit nitride      thick=1 divis=20
etch nitride start x=1.0 y=-10
etch         cont  x=1.0 y=10
etch         cont  x=2.0 y=10
etch         done  x=2.0 y=-10

structure outfile=anelex02_0.str

rate.depo machine=uni oxide a.m  sigma.dep=0.2 \
        uni dep.rate=1000 angle1=0
deposit machine=uni time=1 minute divis=4
structure outfile=anelex02_1.str

init infile=anelex02_0.str
rate.depo machine=uni oxide a.m  sigma.dep=0.2 \
        uni dep.rate=1000 angle1=45
deposit machine=uni time=1 minute  divis=4
structure outfile=anelex02_2.str
...
tonyplot -st anelex02_*.str -ttitle anelex02.in
quit
```

Listing 3.9: Input-Deck

In der ersten Zeile wird der Simulator ausgewählt - in diesem Falle Athena. Die darauf folgenden `line` Anweisungen dienen zur Definition des Gitters für die spätere FEM-Simulation[2]. Hierbei wird das Gitter in den Bereichen, in denen man genauere Ergebnisse haben möchte, enger gewählt. Mit den `deposit nitride` und `etch nitride` Anweisungen wird eine Schicht Siliziumnitrid abgeschieden und strukturiert. Mit der `structure` Anweisung wird die bisherige Schicht als Referenz gespeichert. Danach folgen die Anweisungen für die Abscheidung mit unterschiedlichen Winkeln und zur Ausgabe mit TonyPlot. Andere Prozesse lassen sich auf ähnliche Weise beschreiben. Eine genauere Beschreibung der einzelnen Parameter befindet sich in den entsprechenden Handbüchern zur Software.

Dieses einfache Beispiel verdeutlicht, dass die Eingabe nicht unbedingt benutzerfreundlich ist. Der Nutzer muss die einzelnen Parameter des Simulators genau kennen, um den Simulationsprozess korrekt zu initialisieren. Tippfehler oder die Wahl des falschen Modelles zur Simulation können die Ergebnisse verfälschen und führen zu unnötigen Iterationen bei der Fehlersuche. Die Eingabesprache verfügt über keine Mechanismen, die Korrektheit und Plausibilität der eingegebene Daten zu überprüfen.

Bewertung

Silvaco hat auf dem Gebiet der Simulation jahrelange Erfahrung. Eine Neuentwicklung der angebotenen Simulatoren wäre deshalb eine Verschwendung von Ressourcen. Auf der anderen Seite ist die Bedienung der Software noch etwas umständlich. Die Eingabe erfolgt über Texteditoren und erinnert an die Erstellung von Batch-Jobs. Dies ist natürlich fehlerträchtig. Durch die Integration einer grafischen Nutzerführung könnte die Bedienbarkeit drastisch verbessert werden.

Die Datenhaltung in Form von Dateien ist nicht mehr zeitgemäß. Wird ein Datensatz - zum Beispiel für ein Material - verändert, so muss er in allen betroffenen Dateien geändert werden. Zwar lässt sich durch das Einfügen vorhandener Dateien (z. B. mit Materialmodellen) eine gewisse Hierarchie aufbauen, dies hat jedoch seine Grenzen.

[2] Simulation mit der Finite Elemente Methode

Im EU Projekt Promenade wird eines der Ziele sein, die Simulatoren von Silvaco in eine Softwareumgebung einzubinden, die eine zeitgemäße Datenhaltung und Datenaufbereitung bietet. Die eigentliche Syntax zum Start der Simulatoren soll dabei vor dem Nutzer verborgen werden.

3.2.4 Coventor

Der große Erfolg der Mikroelektronik beruht auf der Tatsache, dass EDA-Software existiert, die den Entwurf von den frühen Phasen bis hin zum Entwurf des Maskensatzes unterstützt. Einen ähnlichen Ansatz für die Mikrosystemtechnik versucht Coventor [Cov04] mit CoventorWare umzusetzen.

Aufbau und Verwendung

CoventorWare bietet verschiedene Tools, die den Entwurf von Mikrosystemen auf verhaltensnaher Ebene unterstützen. Hierzu kann der Entwickler den Entwurfsgegenstand mit Hilfe von Makromodellen beschreiben. Solche Modelle können einfache mechanische Strukturen - wie zum Beispiel Balken - aber auch komplexe fluidische Strukturen - wie ein Ventil - beschreiben. Durch Zusammensetzen mehrerer parametrisierter Makromodelle entstehen Netzlisten, die den Entwurfsgegenstand beschreiben.

Mit Hilfe von entsprechenden Simulatoren kann bereits mit Hilfe dieser Netzliste das Verhalten simuliert werden. Netzlistensimulatoren sind durch die Verwendung einfacherer Modelle meist um ein Vielfaches schneller als FEM-Simulatoren. Trotzdem sind die Ergebnisse ausreichend, um bereits in einem sehr frühen Stadium die Dimensionen des zu fertigenden Produktes prüfen zu können.

Nachdem die Dimensionen des Entwurfes festgelegt sind, müssen für den Entwurf die Masken- und Prozessdaten eingegeben werden. Abbildung 3.13 zeigt den Prozesseditor von Coventor. Hier kann eine vereinfachte Prozessfolge eingegeben werden. Hilfsschritte wie zum Beispiel die Reinigung oder Annealing werden hierbei nicht berücksichtigt. Auch die Korrektheit und Plausibilität der Prozessfolge wird nicht geprüft.

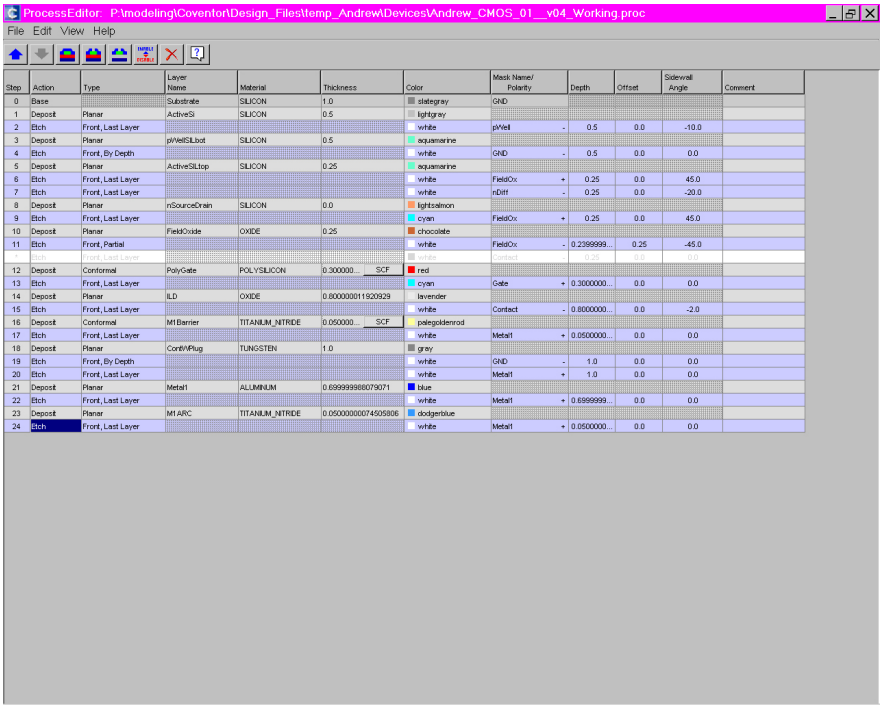


Abbildung 3.13: Prozessschritt Editor von Coventor

Durch Angabe der bei einer Abscheidung entstehenden Materialien, kann danach in Zusammenhang mit einem Maskensatz ein 3-dimensionales Modell erzeugt werden. In Abbildung 3.14 ist das Ergebnis zu sehen. Um die interessanten Schichten (hier Metallkontakte eines Feldeffekttransistors) besser sehen zu können, werden einzelne Schichten entfernt.

Mit Hilfe dieses Modells können nun detaillierte Simulationen mit Hilfe von FEM Software durchgeführt werden. Hier werden zum Beispiel der Stress in einer Schicht oder die Durchbiegung einer Membran simuliert. Die Ergebnisse geben Aufschluss über das reale Verhalten der Komponente. Bei Neuentwürfen können diese Ergebnisse wiederum für die Bildung neuer Makromodellen genutzt werden. Auf diese Weise können die Daten später für neue Entwürfe bereits in früheren Phasen genutzt werden.

Bewertung

Coventor bietet ein umfangreiches Softwarepaket, das den Bereich des verhaltensorientierten Entwurfes [Wag05] abdeckt. Die Fertigung fließt nur in Form

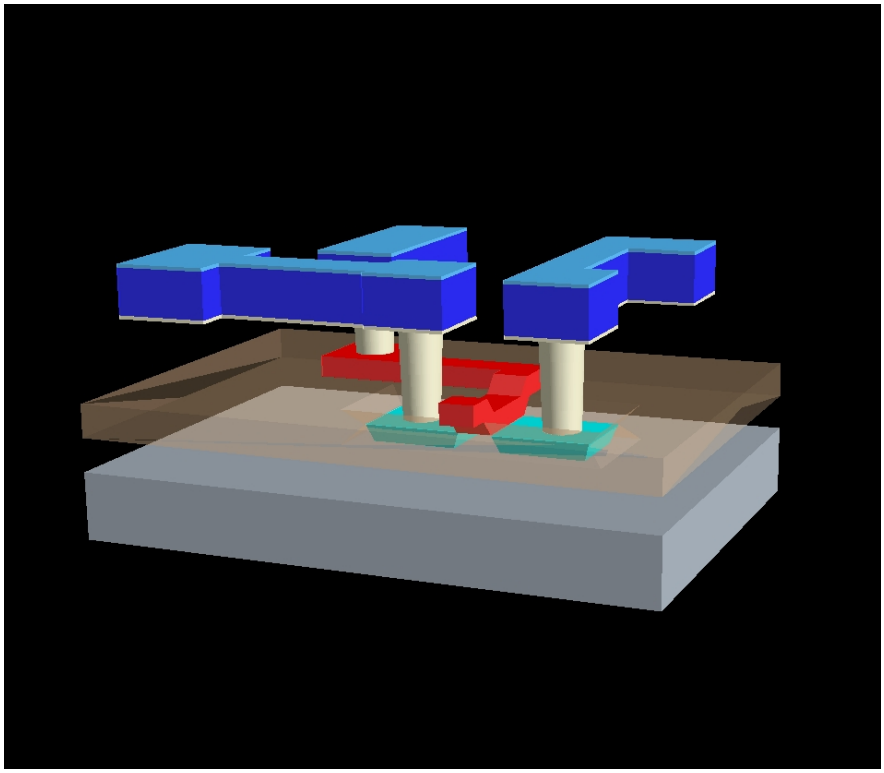


Abbildung 3.14: 3D-Struktur erzeugt mir Coventor

von Material- und Prozessdaten in den Entwurf ein. Die verwendeten Prozessfolgen werden nicht auf Fehler oder Inkonsistenzen geprüft. Deshalb können nur bekannte Prozessfolgen verwendet werden.

Durch Verwendung einer Software, die die Erstellung von Prozessfolgen besser unterstützt, könnten die Ergebnisse verbessert werden. Die benötigten Materialdaten könnten dann „On demand“ durch Simulation erzeugt oder aus der Datenhaltung einer solchen Software geladen werden.

3.2.5 PROMIS

Die Optimierung von Prozessabläufen ist ein wichtiges Thema in der Halbleiterindustrie. Bei steigenden Kosten für die Fertigung von Komponenten in der Mikroelektronik und der Mikrosystemtechnik ist eine optimale Ausnutzung der „Fab“ essentiell für das Überleben des Betreibers. Die Unterstützung der Betreiber bei diesem Bestreben ist das Ziel der Software PROMIS.

Leider stand während der Entstehung dieser Arbeit keine installierte Version des Systems PROMIS zu Testzwecken zur Verfügung. Deshalb beziehen sich die folgenden Aussagen auf die Erfahrungsberichte der Projektpartner beziehungsweise auf Informationen des Herstellers.

Das System PROMIS wird von Brooks Automation [Bro04] vertrieben. PROMIS bietet eine integrierte Software für Planung, Kostenrechnung, Produktions- und Leistungsmanagement. Es bietet Schnittstellen zu den ERP (Enterprise Resource Planning) Systemen und unterstützt die Nutzer bei der Automatisierung der Produktion. Ziel ist es dabei, alle Systeme von der Fab-Automatisierung bis zur Finanzplanung zu integrieren.

Um dies zu ermöglichen, benötigt PROMIS eine zuverlässige Datenhaltung. Hierfür wird das Relationale Datanbankmanagement-System (RDBMS) Oracle (siehe Kapitel 5) verwendet. Durch die Verwendung einer Standard-Datenbank und die Einführung von API's (Application Programmers Interface) können auch andere Produkte an PROMIS gekoppelt werden. So ist es möglich, auch Module zur Steuerung von Maschinen in die Software zu integrieren. Eventuell auftretende Fehler in der Produktion können so schnell entdeckt und behoben werden. Durch die Integration von ERP-Software werden auch die Auswirkungen auf andere Sektoren der Betriebsplanung ersichtlich.

Bewertung

PROMIS ist ein System, das für den Einsatz im Produktionsbereich entwickelt wurde. Die Software PROMIS bietet mächtige Werkzeuge zur Integration und Steuerung. Diese „Mächtigkeit“ ist aber auch gleichzeitig der Punkt, der es für die Erforschung und Entwicklung neuer Systeme nur bedingt nutzbar macht. In Gesprächen mit Entwicklern wurde häufig betont, dass die gebotenen Werkzeuge zu umfangreich und zu komplex wären. Außerdem führt der häufige Wechsel beziehungsweise Umbau des verwendeten Equipments, wie es in der Forschung üblich ist, zu Problemen.

Interessant an PROMIS ist die Verwendung einer professionellen Datenbank. Das Management der anfallenden Daten ist offenbar nur so zu bewältigen. Über die Verwendung der Standardschnittstellen ist es außerdem möglich, mit anderen Werkzeugen die Daten aus der Datenbank zu untersuchen. Hier bietet sich auch ein Anknüpfungspunkt für die in dieser Arbeit vorgestellte Software. Die Übernahme von Werten aus beziehungsweise die Übergabe von Werten an PROMIS oder ähnliche Systeme bieten interessante Perspektiven, die im Rahmen des EU-Projektes Promenade auch untersucht werden.

3.3 Geometriedaten

Neben den Daten, die das Verhalten und die Fertigung eines Mikrosystemes beschreiben, sind auch die Daten, die das Aussehen beschreiben von Bedeutung. Diese Daten können von unterschiedlichster Natur sein.

3.3.1 Maskendaten

Am häufigsten sind die bereits aus der Mikroelektronik bekannten Maskendaten anzutreffen. Hier wird das Aussehen des Produktes durch die Übertragung der Struktur mittels Lithographie erzeugt.

Bereits in den frühen 70ern wurde GDSII[3] von Calma Co.[4] in Sunnyvale, Kalifornien entwickelt. GDSII ist ein binäres Dateiformat, dass sich als Datenaus-

[3] Graphic Design System II
[4] Heute Teil von Cadence

tauschformat für Masken durchgesetzt hat. Es wird hauptsächlich eingesetzt, um die Daten des Designers vom CAD Tool in die eigentliche Fertigungsstätte („Fab“) beziehungsweise zum Maskenhersteller zu übertragen. Dort wird mit Hilfe dieser Datei ein Maskensatz erstellt.

Das Hauptproblem des GDSII Formates ist, dass es bereits in die Jahre gekommen ist. So kann eine GDSII Datei für moderne Designs leicht 50GB erreichen. Außerdem werden nur eine sehr begrenzte Anzahl von Strukturen unterstützt. Kreis- und Bogenstrukturen lassen sich zum Beispiel nur durch Polygone darstellen. Mit OASIS[5] laufen deshalb zur Zeit Bestrebungen der SEMI[6] [Sem04] einen Nachfolger für GDSII zu schaffen. Wann sich dieser Standard durchsetzt, ist jedoch noch nicht abzusehen.

Bewertung

GDSII ist (neben dem bereits vorgestellten Open Access) immer noch der Industriestandard zur Speicherung und Übertragung von Maskendaten. Eine Neuentwicklung eines Datenformates wäre im Rahmen dieser Arbeit nicht sinnvoll. Deshalb sollte nur sicher gestellt werden, dass diese Dateien, welches Format auch immer sie haben, gespeichert werden können. Jedoch sollte bei der Speicherung dieser Daten darauf geachtet werden, dass Metainformationen über den Inhalt der Datei mit gespeichert werden können.

3.3.2 3D Daten

Neben den Verfahren die eine Art der Lithographie verwenden, um Strukturen zu Übertragen, gibt es auch andere Techniken zur Erzeugung einer Struktur. Hierbei kann es sich zum Beispiel um spezielle Lasertechniken handeln, mit denen direkt 3-dimensionale Strukturen aus einem Polymer erzeugt werden können.

Diese Daten sind meist sehr maschinenspezifisch. Es gibt verschiedenste Formate, die meist von den Herstellern der CAD Tools entwickelt wurden. Als

[5] Open Artwork System Interchange Standard
[6] Semiconductor, MEMS and FDP Industry

Beispiel sei hier nur das DXF Format von AutoCAD [Aut04] genannt. Autodesk ist einer der führenden Hersteller für CAD Produkte und das AutoCAD-Format ist mittlerweile so verbreitet, dass viele andere Produkte dieses Format verwenden oder zumindestens importieren können. Jedoch bietet auch weiterhin jeder Hersteller eigene Dateiformate.

Bewertung

Da es viele Formate für die Darstellung eines 3-dimensionalen Gegenstandes gibt, ist es schwierig eine allgemeine Formalisierung für diese zu finden. Die Formate sind zumeist abhängig von der verwendeten Software. Deshalb müssen diese Daten so verwaltet werden, wie sie zur Verfügung stehen. Eine Erweiterung der Dateien um Metadaten, die den Inhalt beschreiben, könnte jedoch die Suche nach entsprechenden Daten erleichtern.

3.3.3 STEP

Bereits 1979 wurde mit IGES (Initial Graphics Exchange Specification) ein Standard zur Archivierung, Speicherung und zum Austausch von technischen Zeichnungen vorgestellt. Im Laufe der Zeit wurde dieses Format von verschiedenen Organisationen weiterentwickelt und für die entsprechenden Bedürfnisse angepasst. Mit der zunehmenden Internationalisierung der Märkte wurde es aber immer dringender notwendig, einen einheitlichen Standard für die Beschreibung von technischen Produkten zu entwickeln. Deshalb wurde von der ISO[7] beschlossen, eine Normenreihe hierfür zu entwickeln.

STEP ist eine Abkürzung für „*ST*andard for *E*xchange of *P*roduct module data“ und ist bei der ISO als Normenreihe ISO 10303 zu finden. Derzeit liegt der Schwerpunkt bei der Anwendung von STEP im Bereich Geometriedatenaustausch [AT00]. Zusätzlich können jedoch auch noch andere Daten, wie zum Beispiel Farbe und Preis übertragen werden. STEP bietet außerdem die Möglichkeit, Informationen über zusammengesetzte Baugruppen zu übertragen.

[7] International Organization for Standardization

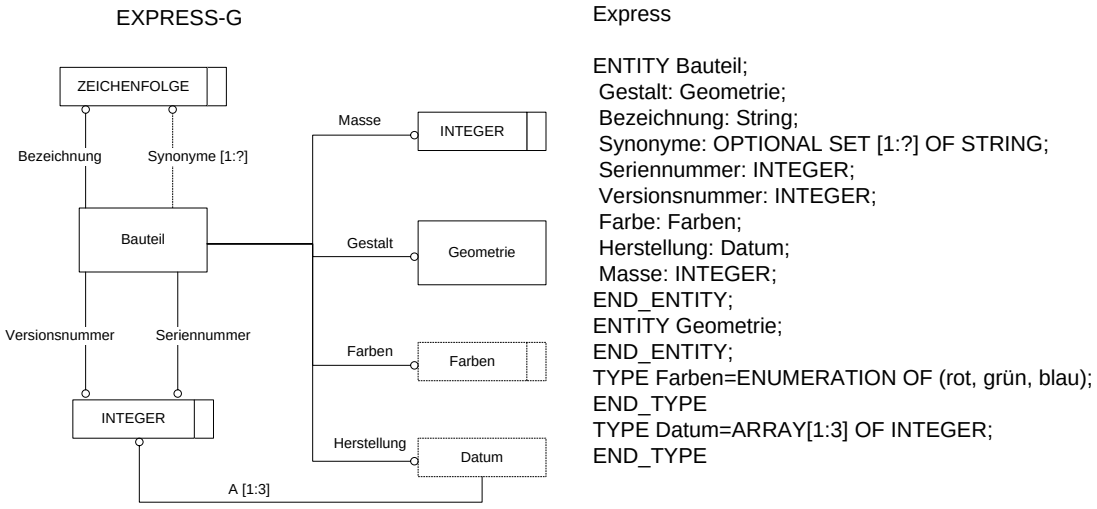


Abbildung 3.15: Notationen von EXPRESS [AT00]

Zur Modellierung wird bei STEP die Sprache EXPRESS verwendet. EXPRESS besitzt dabei sowohl eine textuelle als auch eine grafische Notation (EXPRESS-G). Abbildung 3.15 zeigt ein Beispiel für eine grafische Notation in EXPRESS-G und die entsprechende textuelle Umsetzung in EXPRESS.

Bewertung

Als ISO Standard bietet STEP eine interessante Alternative zur Speicherung und Übertragung von Produktdaten. Es ist zu überlegen, ob man in eine Datenhaltung ein Exportfilter für EXPRESS integriert. Die Spezifikationen in EXPRESS sind jedoch nur auf eine sehr abstrakten Ebene angesiedelt, so dass ein eigener Dialekt mit entsprechender Spezifikation entwickelt werden müsste.

3.4 Fazit

Die in diesem Kapitel vorgestellten Softwareprodukte und Standards beinhalten viele gute Ideen. Besonders für die fertigungsnahen Aspekte fehlt jedoch ein einigendes Framework. Während für die Mikroelektronik hier mit Open Access eine Entwicklung in die richtige Richtung stattfindet, sind bei den in

der Mikrosystemtechnik verwendeten Software Tools meist immer noch proprietäre Dateiformate im Einsatz. Eine gemeinsame, standardisierte Datenhaltung für die Aspekte des fertigungsnahen Entwurfes ist jedoch dringend nötig. Jede der vorgestellten (kommerziellen) EDA-Software Produkte kann durch die Integration der Ergebnisse in eine gemeinsame Datenhaltung profitieren.

Mit den in diesem Abschnitt vorgestellten Software Tools wurden einige interessante Konzepte eingeführt. Die Vererbung als Mittel zur Hierarchisierung und die Verwendung von Datenbanken sind Methoden, die in den folgenden Kapiteln noch näher untersucht werden.

4 Der Entwurfsprozess in der Mikrosystemtechnik

Wenn man sich mit der Haltung der Daten in der Mikrosystemtechnik beschäftigt, muss man sich zwangsläufig auch mit dem Vorgehen der Entwickler beim Entwurf auseinandersetzen. Das Vorgehen beim Entwurf soll im Folgenden als *Entwurfsmethode* bezeichnet werden. Wenn eine Menge solcher Methoden allgemeingültig beschrieben werden kann, so spricht man von einem *Entwurfsmodell*.

In diesem Kapitel sollen die Unterschiedlichen Strategien beim Entwurf kurz beleuchtet und auf ihre Relevanz für die Datenhaltung untersucht werden. Eine genauere Untersuchung der Entwurfsmethoden und Modelle in der Mikrosystemtechnik kann in [Wag05] gefunden werden.

4.1 Vorgehen beim Entwurf

Je nach Ausgangspunkt und Ziel eines Entwurfes gibt es unterschiedliche Vorgehensweisen. Man unterscheidet hierbei drei unterschiedliche Prinzipien. Diese sind bereits aus der Mikroelektronik bekannt, müssen jedoch in der Mikrosystemtechnik in einem anderen Kontext gesehen werden:

Bottom up: Beschreibt ein Entwurfsvorgehen, bei dem mit den technologienahen Entwurfsschritten begonnen wird. Mittels vorhandener oder neu zu entwickelnder Prozessschritte wird ein neues Produkt entwickelt. Die Funktion und das Verhalten wird erst während des Entwurfsprozesses durch die gewählten Materialien und Prozessschritte festgelegt.

Dieses Vorgehen findet vor allem bei der Entwicklung neuer Technologien Einsatz. Meistens wird hier mit einer wagen Idee begonnen die in

iterativen Schritten mit Hilfe vieler praktischer Versuche verfeinert wird. Als Ergebnis solcher Vorgehensweisen stehen am Ende meist Technologiedemonstratoren zur Verfügung.

Top down: Beschreibt ein Entwurfsvorgehen, bei dem mit der Spezifikation der Funktion und des Verhaltens begonnen wird. Bereits zu Beginn werden hier umfangreiche Simulationen durchgeführt, um das spätere Verhalten des Systems vorherzusagen.

Im weiteren Entwurfsverlauf wird dann die geometrische Struktur generiert, mit der das Verhalten umgesetzt werden kann. Schließlich wird die Produktionsanweisung generiert. Hierbei wird eine Folge von Prozessschritten gesucht, bei der ein der gewünschten Struktur möglichst ähnliches Ergebnis erhalten wird.

Besonders der letzte Schritt ist bis heute fast nicht automatisierbar. Hier ist man auf die Erfahrung des Designers angewiesen. Neuere Ansätze versuchen diese Problematik durch Verwendung von Bibliotheken mit vorgefertigten Prozesssequenzen und Strukturen zu umgehen. Dies schränkt jedoch die Möglichkeiten des Entwurfsprozesses stark ein.

Meet in the middle: Ist eine Vereinigung beider Ansätze. Das Bottom up Vorgehen wird verwendet, um Prozessfolgen für Strukturen wie Schichten oder Hebel zu entwerfen. Die Ergebnisse dieser Entwürfe werden in Bibliotheken gespeichert. Auf der anderen Seite kann mittels des Top down Ansatzes das Verhalten und die Funktion spezifiziert werden. Durch mehr oder weniger automatisierte Generierungsschritte kann eine geometrische Struktur entwickelt werden.

Mit Hilfe der Bibliotheken muss dann eine geeignete Prozessfolge zur Herstellung dieser Struktur gesucht werden. Der letzte Schritt ist dann der Entwurf geeigneter Masken (oder anderer geometrischer Anweisungen), um die gewünschte Struktur zu fertigen.

Abbildung 4.1 zeigt schematisch das aktuell umgesetzte Entwurfsvorgehen im fertigungsnahen Entwurf. Auf den ersten Blick ist hier ein klassisches Bottom-up Schema umgesetzt. Jedoch wird im Punkt *Funktionalen Schicht ermitteln* sehr viel Wissen aus dem Top down Entwurf benötigt. Die funktionale Schicht ist die Schicht, die die Wirkungsweise bestimmt. Sie kann zum Beispiel die auf

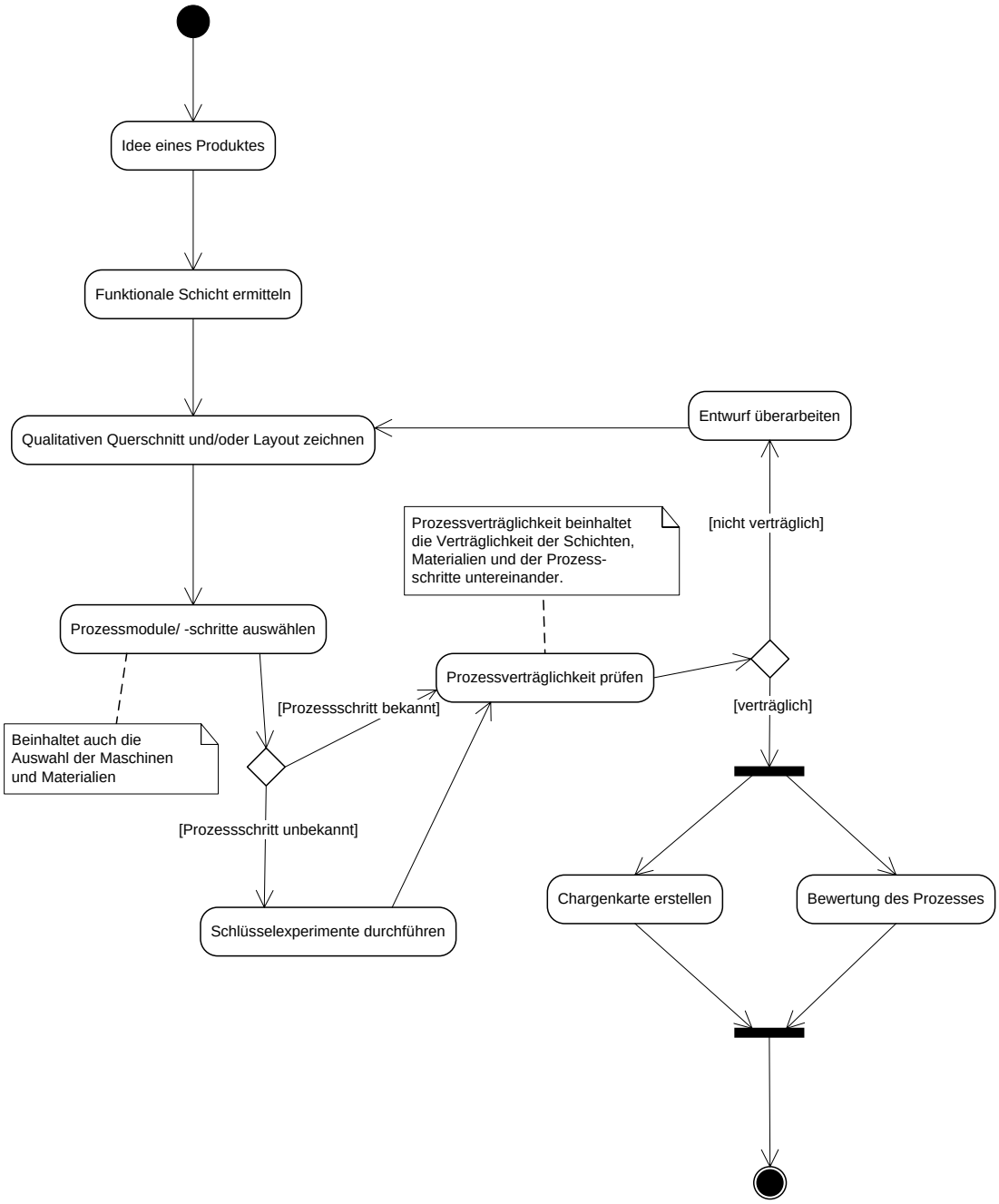


Abbildung 4.1: Produktentwicklung [Wag01]

Druckänderungen sensitive Membran oder der Hebel, der Beschleunigungen misst, sein. Die Anforderungen an Verhalten und Funktion bestimmen wesentlich die Art und den Aufbau der funktionalen Schicht.

Auf der anderen Seite sind die Daten, die durch die *Schlüsselexperimente* gewonnen werden, von großer Bedeutung für die Schnittstelle zwischen Top down und Bottom up Entwurf. Hier werden Daten gesammelt, die das reale Verhalten beschreiben und so zukünftig einen Entwurf auf höherer Ebene ermöglichen. Es ist also zu erkennen, dass eine konsistente und durchgängige Datenhaltung das aktuelle Entwurfsvorgehen durchaus beschleunigend unterstützen kann.

4.2 Entwurfsmodelle

Um den Entwurf zu automatisieren muss versucht werden, das Entwurfsvorgehen in einer allgemeinen, abstrakten Form zu beschreiben. Entwurfsmodelle stellen solche Beschreibungen dar. Im folgenden werden einige dieser Modelle vorgestellt.

4.2.1 Das Y-Modell

Da die Mikrosystemtechnik bereits die Mikroelektronik beinhaltet, macht es natürlich Sinn, sich mit den dort verwendeten Modellen zu beschäftigen. Hierbei fällt auf, dass sich zumindest in der digitalen Mikroelektronik das Y-Modell [GK83, WT85] durchgesetzt hat. Abbildung 4.2 zeigt eine Darstellung dieses Modells.

Das Y-Modell beschreibt den Entwurf von digitalen Schaltungen in der Mikroelektronik beginnend mit der abstrakten, meist mathematischen Beschreibung des Entwurfsgegenstandes. Über mehrere inzwischen meist automatisierte Generierungsschritte wird danach das System bis auf die Maskenebene spezifiziert.

Dieses Modell lässt sich jedoch leider nicht auf die Mikrosystemtechnik anwenden. Das Hauptproblem ist, dass das Y-Modell mit der Spezifikation der Masken endet. Der Grund hierfür ist die Trennung des Entwurfes der Fertigung

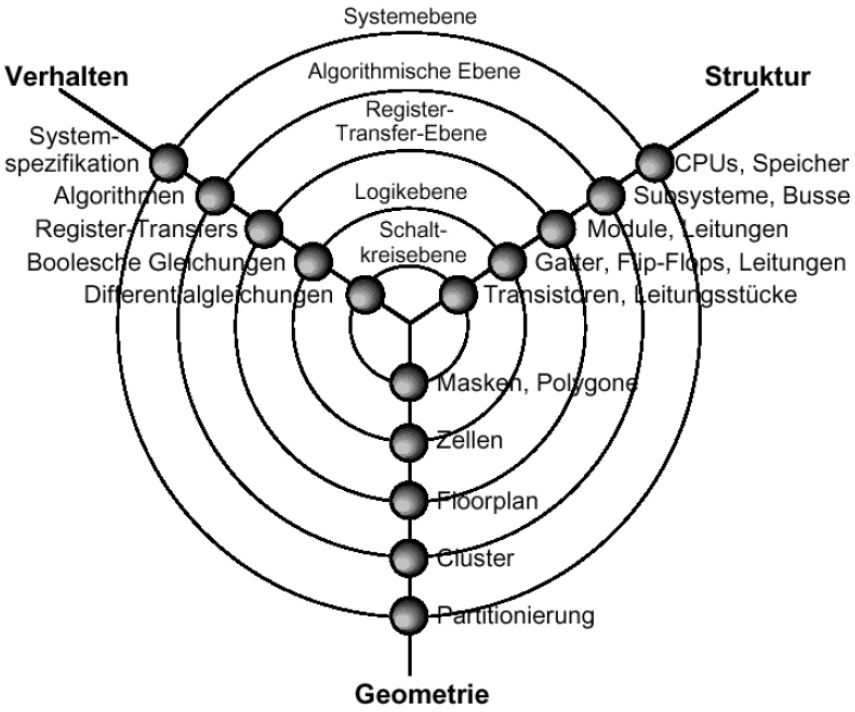


Abbildung 4.2: Das Y-Modell [GK83, WT85]

vom Entwurf der Schaltung (beschrieben und gefordert in [MC80]). Dies ist möglich, indem die Prozessfolge zur Fertigung durch eine festgelegte Schnittstelle - die Layout Regeln - beschrieben wird. Diese Regeln sind ausreichend, um die Einflüsse der physikalischen Fertigung auf den Entwurf der Schaltung zu formalisieren.

Während in der Mikroelektronik die zwei Dimensionen des Maskenentwurfs ausreichend sind, um eine Schaltung zu beschreiben, ist dies in der Mikrosystemtechnik nicht der Fall. Die dritte Dimension hat eine entscheidende Bedeutung. Da mit mechanischen Strukturen gearbeitet wird, werden das Verhalten und die Eigenschaften dieser Strukturen durch den vertikalen Aufbau der Schichten beeinflusst. Somit wird die Auswahl und Anpassung der Fertigungsfolge zum integralen Bestandteil des Entwurfs. Abbildung 4.3 stellt die Problematik dar. Während in der Mikroelektronik der Einfluss der Fertigung auf den Entwurf in Layout-Regeln abstrahiert werden kann, ist dies in der Mikrosystemtechnik nicht möglich. Die Art der Fertigung hat Einfluss auf alle Entwurfsschritte bis hin zur Auswahl des Funktionsprinzips.

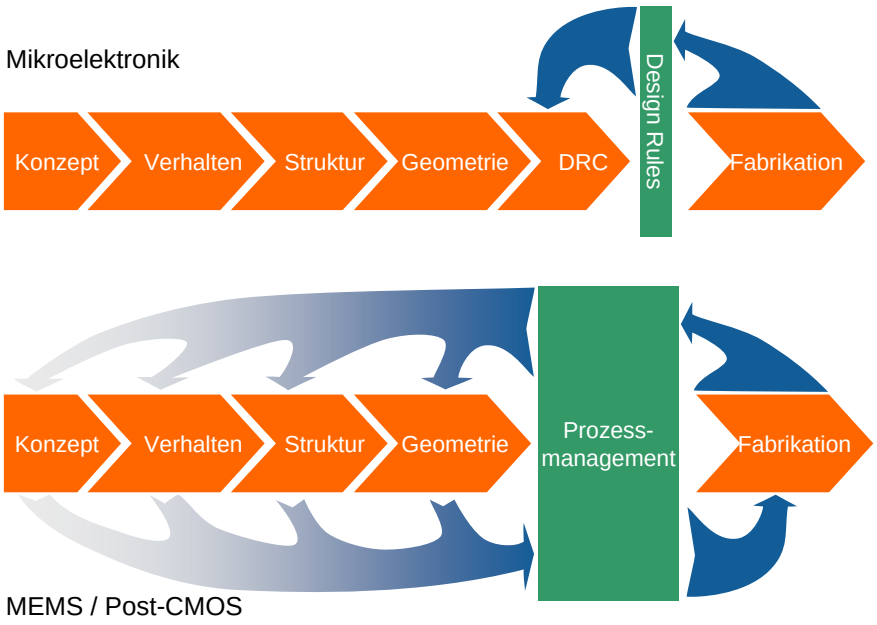


Abbildung 4.3: Mikrosystemtechnik und Mikroelektronik [HW03, Wag05]

Auch in der Mikroelektronik wird jedoch der Einfluss der Fertigung auf die frühen Entwurfsphasen immer gravierender. Mit Entwicklung immer komplexerer Technologien und Verwendung immer feinerer Strukturen wird sich dies in den nächsten Jahren noch verschärfen. Deshalb wird sowohl für die Mikroelektronik als auch für die Mikrosystemtechnik ein Technologiemanagement, das die Verwaltung der Fertigungsdaten beinhaltet, immer dringender nötig.

4.2.2 Das Kreismodell

Die im vorigen Abschnitt beschriebene Problematik wurde bereits vor mehreren Jahren erkannt. In [Brü96, Hah98a] wurde das Kreismodell vorgestellt, welches im Verlauf des PRINCE-Projektes verfeinert wurde (siehe Abbildung 4.4). Das Kreismodell beschäftigt sich mit den fertigungsnahen Phasen des Mikrosystementwurfes. Eingabe in dieses Modell ist eine Fertigungsspezifikation. In mehreren iterativen Schritten wird dann aus dieser Beschreibung ein Maskensatz und eine Fertigungsanweisung erstellt.

Das Interessante an diesem Modell ist, dass die generierenden und überprüfenden Aktivitäten auf ein Repository zurückgreifen (in Abbildung 4.4 in der

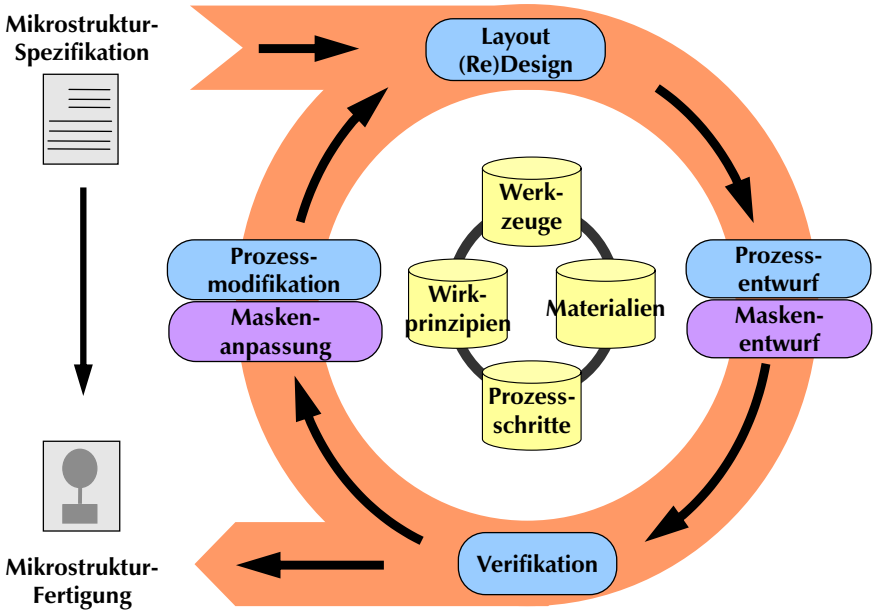


Abbildung 4.4: Erweitertes Kreismodell

Mitte dargestellt). In diesem Repository sind Daten über Materialien, Prozessschritte, Wechselwirkungen, Wirkprinzipien und Maschinen. Das heißt, bereits in diesem Modell wird die Notwendigkeit einer umfassenden Datenhaltung bestätigt.

In [Sch04b] wird ein mehrstufiges Kreismodell vorgestellt. Hier werden auch die Aspekte des Verhaltens und der Struktur berücksichtigt. Auch in diesem Modell ist die Datenhaltung zentraler Punkt aller Aktivitäten.

4.2.3 Allgemeines Entwurfsmodell

Während der Arbeit am PRINCE System wurde am Institut ein allgemeines Entwurfsmodell entwickelt. Hierbei werden sowohl der fertigungsnahe (bottom-up) als auch der verhaltensorientierte (top-down) Entwurf berücksichtigt. Abbildung 4.5 zeigt eine schematische Darstellung dieses Modelles.

Im unteren Teil ist zu erkennen, dass beide Arten des Entwurfsvorgehen mit einer Spezifikation der Anforderungen beginnen. Hier wird festgelegt, nach welchem Prinzip das Produkt funktionieren soll. Danach unterscheiden sich jedoch die Vorgehen.

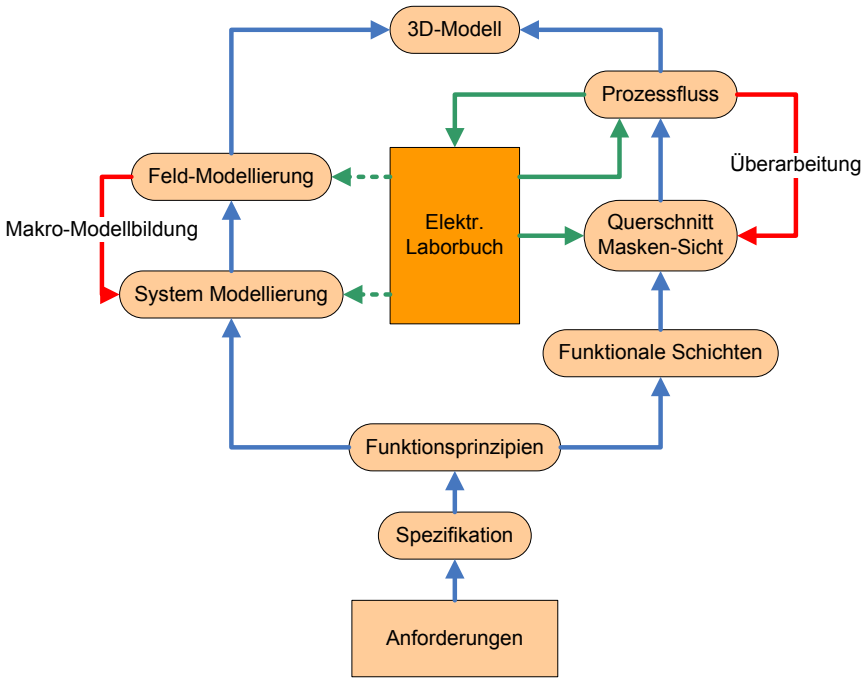


Abbildung 4.5: Allgemeines Entwurfsmodell [PWOB04]

Im rechten Zweig des Modelles wird der fertigungsnahe Entwurf modelliert. Hierbei wird nach der Festlegung einer funtkionalen Schicht und einer prinzipiellen Struktur eine passende Prozessfolge zur Fertigung gesucht. Dazu werden Experimente und vorhandene Daten zur Findung verwendet. Das dabei genutzte iterative Vorgehen ist im Kreismodell beschrieben.

Der linke Zweig beschreibt den verhaltensorientierten Entwurf. Hier wird mittels FEM und Netzlistensimulation das Verhalten von möglichen Strukturen getestet, bevor eine endgültige Struktur entsteht. Auch hierzu wird ein iterativer Prozess verwendet.

Das interessante an diesem Modell ist die Verwendung eine Wissensdatenbank. Hier werden alle Informationen über den Entwurf gespeichert. Daten die mittels Experiment im fertigungsnahem Entwurf gewonnen wurden, können so der Simulation im verhaltensorientierten Entwurf zu Gute kommen. Je besser die Datenbasis ist, desto später muss mit realen Experimenten begonnen werden. Auf diese Weise kann Zeit und Geld beim Entwurf gespart werden.

Bei genauerer Betrachtung werden hier noch einige weitere Anforderungen an die Datenhaltung deutlich. So werden im fertigungsnahem Entwurf viele Experimente durchgeführt, die sich nur in kleinen Parameteränderungen unterscheiden. Hier wäre ein Mechanismus, der Schablonen bereitstellt sehr vorteilhaft.

Ein weiterer zu berücksichtigender Punkt sind die frühen Entwurfsphasen. Hier weiß der Entwickler in der Regel noch nicht, welche Prozessschritte er genau einsetzen möchte. Es sind lediglich Daten über gewünschte Eigenschaften von Schichten oder Prozessschritten vorhanden. Deshalb muss eine Möglichkeit bereitgestellt werden, mit solchen „abstrakten“ Prozessdaten umzugehen. Im weiteren Verlauf können diese Daten dann immer genauer spezifiziert werden, bis ein eindeutiges Rezept angegeben werden kann. Hierbei muss berücksichtigt werden, dass das gleiche Rezept durchaus auf unterschiedlichen Wegen erreicht werden kann. Ein Prozessschritt, der beispielsweise eine Siliziumoxid-Schicht abscheidet, kann sowohl als Oxidabscheidung als auch als Opferschichtabscheidung klassifiziert werden.

All diese Anforderungen erinnern sehr stark an Problematiken, mit denen sich der objektorientierte Entwurf beschäftigt. Schablonen und abstrakte Daten kann man durchaus als Klassen bezeichnen, und der Entwurfsverlauf mit einer immer genaueren Spezifikation der Daten kann als Spezialisierung in einer Vererbungshierarchie verstanden werden. Schließlich bietet die Mehrfachvererbung einen Mechanismus, mit dem das gleiche Objekt unterschiedlichen Bedeutungskreisen zugeordnet werden kann. Kapitel 6 beschäftigt sich eingehend mit der Umsetzung dieser Analogie.

4.3 Entwurfsaufgaben

In [Sch04b] wird der Entwurf in Entwurfsaufgaben unterteilt. Eine Entwurfaufgabe beschreibt hierbei einen Schritt im Entwurf. Einige dieser Entwurfsaufgaben sind bereits automatisiert, die meisten werden jedoch noch manuell oder nur mit geringer Softwareunterstützung durchgeführt. Die meisten Entwurfsaufgaben benötigen sehr viele Daten, um die entsprechende Aufgabe erfolgreich zu lösen. Im folgenden sollen einige dieser Entwurfsaufgaben

aus dem fertigungsnahen Entwurf vorgestellt und die Anforderungen an die Datenhaltung untersucht werden.

4.3.1 Generierung von Prozessfolgen

Selbst beim fertigungsnahen Entwurf beginnt der Designer nicht sofort mit Experimenten. Normalerweise steht am Beginn ein Prinzipentwurf, aus dem die Struktur des Produktes hervorgeht. Vor allem die funktionale Schicht wird hier spezifiziert. Die Entwurfsaufgabe ist hier die Generierung einer fertigbaren Prozessfolge aus diesem Prinzipentwurf. Das Tool Mistic (Siehe Kapitel 3.1.2) ist ein Ansatz für die automatische Durchführung dieser Entwurfsaufgabe. Die Komplexität dieser Aufgabe ist jedoch enorm.

Als erstes gilt es die richtigen Materialien zu finden. Häufig werden für die Materialparameter nur sehr unspezifische Angaben gemacht. So kann zum Beispiel für manche Materialien nur angegeben werden, dass für den Leitwert gewisse Grenzen eingehalten werden müssen. Dabei ist wiederum zu beachten, dass das gleiche Material auf unterschiedliche Art und Weise charakterisiert werden kann. Auch hier drängt sich also wieder die Analogie mit der Objektorientierung auf.

Nachdem die Materialien genau(er) spezifiziert sind, muss man geeignete Prozessschritte finden, mit denen Schichten aus diesem Materialien herstellbar sind. Dabei müssen Einschränkungen durch bereits abgeschiedene Materialien bzw. bereits ausgeführte Prozessschritte berücksichtigt werden. Außerdem muss das Material eventuell noch strukturiert werden. Hierzu muss eine geeignete Methode gefunden werden, die nur das gewünschte Material (bzw. die gewünschten Materialien) nicht aber unbeteiligte Schichten angreift.

Nach dem Abschluss dieser Schritte hat man eine prinzipielle Prozessfolge zur Erstellung des gewünschten Fertigungsgegenstandes. Diese ist jedoch noch nicht ausreichend. So müssen diverse Hilfsschritte, wie Reinigung, das Aufbringen von Haftschichten oder Annealing eingefügt werden, um eine fertigbare Prozessfolge zu erstellen.

Vom Standpunkt der Datenhaltung ergeben sich hier viele Problemstellungen. Um diese Entwurfsaufgabe erfolgreich durchzuführen müssen Daten über

Materialien, Prozessschritte und Wechselwirkungen (Haftung, Ätzselektivität, Hilfsschritte) bereitgestellt werden. Mit der Bereitstellung der Daten allein ist es jedoch nicht getan. Es müssen geeignete Mechanismen zum Suchen und zur Kategorisierung der Daten verfügbar sein. Hierbei muss berücksichtigt werden, dass es mehrere Wege zum Kategorisieren der Daten gibt, die alle sinnvoll sein können.

4.3.2 Konsistenzcheck

Im vorigen Abschnitt wurde erwähnt, dass verschiedene Prozessschritte oder Materialien nicht miteinander verträglich sind bzw. einander erfordern. Eine Prozessfolge kann als *konsistent* bezeichnet werden, wenn alle diese Abhängigkeiten beachtet werden. Der Begriff der *Konsistenz* einer Prozessfolge garantiert lediglich, dass eine Prozessfolge fertigbar ist. Er sagt nichts über die Eigenschaften der Produkte aus, die auf Basis dieser Prozessfolge gefertigt werden.

Eine automatische Überprüfung der Konsistenz von Prozessfolgen kann also erheblich zur Einsparung finanzieller und zeitlicher Ressourcen dienen. Daraus leitet sich die Entwurfsaufgabe des Konsistenchecks ab. Der Konsistenzcheck überprüft (automatisch), ob eine Prozessfolge konsistent ist. Eine erste Umsetzung für eine automatische Durchführung dieser Entwurfsaufgabe wurde mit Lido (siehe Kapitel 3.1.3) vorgestellt. Hierbei wurde bereits erkannt, dass eine Automatisierung nur möglich ist, wenn die Abhängigkeiten in Regeln formalisiert und in einer rechnergerechten Form aufbereitet und gespeichert sind. Bei einer genaueren Analyse des Problemfeldes wurden verschiedene Typen von Regeln erkannt. Im Folgendem sollen diese Regeln genauer vorgestellt werden.

Prozessschrittabhängige Regeln

Dieser Regeltyp wurde bereits in Lido behandelt. Die Regeln legen eine bestimmte Art der Vor- oder Nachbehandlung von Prozessschritten fest. Beispiele hierfür sind ein Reinigungsschritt, der vor einer Deposition stattfinden muss oder ein Annealing Schritt nach der Deposition, um den Stress in der abgeschiedenen Schicht zu verringern. Die Vor- oder Nachbehandlung ist jedoch

nicht nur auf Prozessschritte beschränkt, sondern kann auch durch Parametereinschränkungen ausgedrückt werden. So kann zum Beispiel ein Annealing Schritt oft weggelassen werden, wenn ein anderer Prozessschritt die nötige Temperatur erreicht. In so einem Fall reicht die Angabe einer Mindesttemperatur.

Ein weiterer zu berücksichtigender Punkt ist die Begrenzung der Gültigkeit bestimmter Regeln. Oft sind Regeln nur unter bestimmten Bedingungen oder eine gewisse Zeitspanne gültig. Auch dies muss dargestellt werden können.

Schließlich fällt auch hier auf, dass es Regeln gibt, die für eine ganze Gruppe von Prozessschritten gilt (z. B. eine Reinigung sollte vor jeder Deposition stattfinden). In solchen Fällen ist es sinnvoll, diese Regel an die Gruppe zu binden und allen Prozessschritten dieser Gruppe zur Verfügung zu stellen. Einen solchen Mechanismus stellt wiederum die Objektorientierung durch die Vererbung zur Verfügung.

Maschinenabhängige Regeln

Maschinenabhängige Regeln sind prinzipiell ähnlich aufgebaut wie prozessschrittabhängige. Es handelt sich hier lediglich um Einschränkungen, die durch Maschinen zur Fertigung verursacht werden. Ein Beispiel hierfür ist die Reinheitsklasse einer Maschine. Prinzipiell dürfen nur Werkstücke nur in Maschinen mit gleicher oder aufsteigender[1] Reinheitsklasse bearbeitet werden. Wird dies nicht beachtet, so wird die entsprechende Maschine verunreinigt und die Reinheitsklasse dieser Maschine muss angehoben werden.

Materialspezifische Regeln

Diese Regeln beschreiben Einschränkungen, die durch die Eigenschaften des Materials hervorgerufen werden. Ein Beispiel hierfür ist der Schmelzpunkt. Wird ein Werkstück über den Schmelzpunkt erhitzt, so wird jede Schicht, die aus dem betreffenden Material besteht, zerstört oder zumindest beschädigt.

[1] Je nach Hersteller unterscheidet sich unter Umständen die Semantik. Mit aufsteigend ist hier ein höherer Verschmutzungsgrad gemeint.

Materialspezifische Regeln lassen sich nur dann prüfen, wenn Daten über die verwendeten Materialien vorhanden sind. Hier findet sich also ein weiterer Punkt für die Notwendigkeit einer Materialdatenbank.

Geometrie und Layout Regeln

Die Geometrie betreffende Regeln lassen sich nur sehr schwer formalisieren. Zwar lassen sich - wie in Lido geschehen - allgemeine, durch die Lithographie oder Ätzschritte hervorgerufene Strukturgrößen speichern und überprüfen. Die meisten dieser Regeln sind jedoch von der Intention des Nutzers abhängig. So ist zum Beispiel eine Schutzdotierung um ein Gebiet nur dann sinnvoll, wenn es für die Funktion des Entwurfsgegenstandes entscheidend ist.

Ein Konsistenzcheck ist nur dann möglich, wenn alle Regeln in geeigneter Art und Weise in einer Datenhaltung erfasst werden. Es müssen Mechanismen bereitgestellt werden, die es erlauben Regeln an ganze Gruppen von Prozessschritten oder Materialien zu binden. Auch hier spielt also die Datenhaltung eine bedeutende Rolle für die Durchführung der Entwurfsaufgabe.

4.3.3 Optimierung

Die Entwurfsaufgabe Optimierung beschäftigt sich mit der Verbesserung vorhandener Prozessfolgen. Eine automatisierte Durchführung der Optimierung kann zur erheblichen Kostenersparnis führen. Die Software kann bei geeigneter Datenhaltung auf Daten aller Entwickler zugreifen. So können neue Ideen und Umsetzungen von Problemlösungen in den Optimierungsprozess einfließen.

Eine Optimierung kann nach unterschiedlichsten Kriterien erfolgen. Einige dieser Kriterien, wie zum Beispiel Kosten, lassen sich relativ leicht umsetzen. Dies kommt daher, dass sich der zu optimierende Parameter - hier die Kosten - direkt ermitteln lässt. Bei der Optimierung von Parametern, die die Funktion betreffen (z. B. Stress), ist dies nicht so leicht möglich. Hier ist eine Optimierung nur möglich, wenn durch empirische Untersuchung ausreichend Daten zur Verfügung stehen.

Zur erfolgreichen Durchführung der Optimierung muss die Datenhaltung spezielle Mechanismen anbieten. Als erste Anforderung ist ein Suchmechanismus zu nennen, der es erlaubt, Prozessschritte oder auch neue Materialien nach frei wählbaren Kriterien zu finden. Dies erhöht die Flexibilität bei der Optimierung einer Prozessfolge. Eine freie Suche ist heute in vielen Datenhaltungssystemen technisch umgesetzt. Lediglich der Entwurf des zugrundeliegenden Datenschemas muss geschickt gewählt werden, um Anfragen nicht unnötig komplex zu gestalten.

Neben dieser freien Suche wird jedoch auch eine hierarchische Suche benötigt. Hier ist das Ziel, nur Prozessschritte zu finden, die dem vorhandenen „ähnlich“ sind. In der Praxis heißt das, dass man Prozessschritte sucht, die in einer hierarchischen Ordnung der Prozessschritte (z. B. Graph oder Baum) relativ nahe beieinander liegen.

Über die bereits mehrmals erwähnte Vererbung in der Objektorientierung lässt sich auch diese Problemstellung lösen. Ähnlich sind prinzipiell alle Prozessschritte oder Materialien die einer Klasse angehören. Auch bei der Optimierung ist also das Vorhandensein einer umfassenden Datenbasis und die geeignete Aufbereitung der Daten von großer Bedeutung. Die Daten müssen schnell verfügbar und variabel im Zugriff sein.

4.3.4 Simulation von Prozessfolgen

Bereits mit SIMPL (siehe Kapitel 3.1.1) wurde ein Tool vorgestellt, das die Daten einer vorhandenen Prozessfolge zur Simulation benutzt. Ziel der Simulation ist es, eine zweidimensionale (Querschnitt) oder dreidimensionale grafische Darstellung des Entwurfsgegenstandes zur erhalten. Außerdem sollen in den meisten Fällen Parameter, wie zum Beispiel der Stressgradient einer Schicht, ermittelt werden.

Jede Simulation kann hierbei unterschiedliche Genauigkeiten haben. Je genauer simuliert werden soll, desto mehr Zeit und Daten werden benötigt. Neben den Daten des Prozessschrittes sind auch Daten über Materialien und deren Wechselwirkungen notwendig. Manchmal sind diese Daten jedoch noch nicht verfügbar. Um die Simulation einer ganzen Prozessfolge dennoch zu

ermöglichen, muss für solche Prozessschritte auf einfachere Simulationsmodelle zurückgegriffen werden.

Neben den unterschiedlichen Genauigkeitsstufen gibt es auch die Möglichkeit, dass unterschiedliche Simulationsverfahren angewendet werden können. Ein Ätzprozess lässt sich beispielsweise sowohl mittels eines zellulären Automaten als auch mit geometrischen Algorithmen simulieren [Sch04a]. Je nach Zielsetzung der Simulation kann das Ergebnis mit dem einen oder dem anderen Verfahren besser sein.

Für die Simulation von Prozessfolgen ist also eine umfangreiche Verwaltung der Daten nötig. Neben den eigentlichen Daten müssen auch Mechanismen bereitgestellt werden, die für einen Prozessschritt unterschiedlich genaue Simulationsmodelle verwalten und nach abstrakteren Modellen suchen, falls keine genauen Modelle verfügbar sind.

4.4 Fazit

Der Entwurfsprozess in der Mikrosystemtechnik ist zur Zeit nur unzureichend automatisiert. Vor allem im fertigungsnahen Bereich ist kaum Software zur Unterstützung des Designers vorhanden. Um dies zu ändern, müssen Werkzeuge bereitgestellt werden, die die verschiedenen Entwurfsaufgaben durchführen. Jedes dieser Werkzeuge ist auf eine umfassende Datenhaltung angewiesen. Dabei ist es wichtig, sich über die Strukturen, in denen die Daten gespeichert werden, genauere Gedanken zu machen.

Hieraus ergeben sich verschiedene Anforderungen an das Datenmanagement. Die erste Anforderung ist natürlich ein schneller Zugriff auf die Daten. Es müssen Möglichkeiten gegeben sein, die Suche zu beschleunigen. Hierzu können Strukturen wie zum Beispiel Indizes angelegt werden. Eine weitere Anforderung sind Hierarchien, in denen ein Datenobjekt auf mehreren Wegen erreicht werden kann. Hierbei wurde erkannt, dass Technologien aus der Objektorientierung angewandt werden können, um dieses Problem zu lösen. Besonderes Augenmerk ist hierbei auf die Mehrfachvererbung zu legen. Aufgrund der durch die Mehrfachvererbung auftretenden Probleme wurde sie in bisherigen Arbeiten (siehe Kapitel 3) noch nicht untersucht.

Zusammenfassend lässt sich sagen, dass eine durchdachte Datenhaltung ein integraler Bestandteil eines jeden zu entwickelnden Softwaresystems zur Unterstützung des Entwurfs in der Mikrosystemtechnik sein muss. Nahezu alle Daten, die beim Entwurf anfallen, sind eng miteinander verknüpft, was in der Datenhaltung zum Ausdruck gebracht werden muss. Die Datenhaltung muss deshalb hohen Anforderungen an Geschwindigkeit und Ausdrucksfähigkeit genügen.

Durch Softwaretools in Zusammenarbeit mit einer umfassenden Datenbasis lässt sich der Entwurf erheblich vereinfachen und beschleunigen. Kosten- und zeitaufwendige Fehler lassen sich so vermeiden. Im folgendem Kapitel werden die technischen Möglichkeiten der Datenhaltung auf ihre Eignung untersucht.

5 Datenbanken

In den vorhergehenden Kapiteln wurde deutlich, dass im fertigungsnahen Entwurf viele Daten anfallen. All diese Daten sind eng miteinander verknüpft und haben komplexe Beziehungen zueinander. Deshalb müssen diese Daten in geeigneter Weise verwaltet werden. In diesem Kapitel sollen aktuelle Systeme zur Verwaltung von Daten vorgestellt und auf ihre Tauglichkeit für die Problemstellungen im fertigungsnahen Entwurf von Mikrostrukturen untersucht werden.

5.1 Was ist eine Datenbank?

Vor allem in den älteren Arbeiten werden Begriffe wie Datenbank oder Datenbanksystem recht freizügig gebraucht. Deshalb soll zu Beginn eine kurze Begriffsklärung stattfinden.

Datenbanksystem: Ein Datenbanksystem ist ein rechnergestütztes Informationssystem mit der Eigenschaft, dass die Daten des Systems allgemein verfügbar sind [Rol03].

Besonders der Begriff „allgemein“ ist in dieser Beschreibung zu beachten. Allgemein heißt in diesem Zusammenhang, dass die Daten von einer Vielzahl von Anwendungen zu verarbeiten sind. Deshalb dürfen die Daten nicht in einer Form vorliegen, die nur von einer Anwendung verstanden wird.

5.1.1 Verwaltung von Daten in Dateien

Die Mehrzahl der in Kapitel 3 vorgestellten Programme verwenden Dateien als einfach umzusetzendes Mittel zur Speicherung von Daten. Die Verwaltung

von Daten in einem Dateisystem hat jedoch viele Nachteile. Dies sind unter anderem

- Hohe Redundanz
- Keine Backuplösung
- Keine Mehrbenutzerfähigkeit

Während die Mehrbenutzerfähigkeit durch Sperrmechanismen des Betriebssystems und das Backup durch andere Software rudimentär sichergestellt werden können, ist die Redundanz der Daten ein echtes Problem. Redundanz bezeichnet die Tatsache, dass die gleichen Daten in unterschiedlichen Dateien gespeichert sein können. Dies ist aus mehreren Gründen gefährlich:

Mehrdeutigkeit: identische Objekte können in verschiedenen Dateien unterschiedlich benannt sein. Dies erschwert die Verknüpfung und Auswertung von Daten.

Inkonsistenz: wenn mehrere Instanzen von einem Objekt existieren, muss sichergestellt sein, dass bei einer Änderung alle Instanzen angepasst werden. Dies kann das Dateisystem nicht leisten.

Mehraufwand: wenn Daten in unterschiedlichen Dateien gehalten werden, so muss bei einer Änderung jede Datei geändert werden, in der die Daten enthalten sind.

Diese Probleme können vermieden werden, wenn man die Daten der einzelnen Systeme in einem Datenbanksystem zusammenfasst. Verschiedene Anwendungen können auf einen einheitlichen Datenbestand zugreifen und so die Redundanz beseitigen.

5.1.2 Aufbau und Aufgaben eines Datenbanksystems

Ein Datenbanksystem erreicht die Unabhängigkeit der Daten von den Anwendungen, indem den einzelnen Anwendungen die Verantwortung für das Datenmanagement abgenommen wird. Diese Aufgabe wird an eine tiefer liegen-

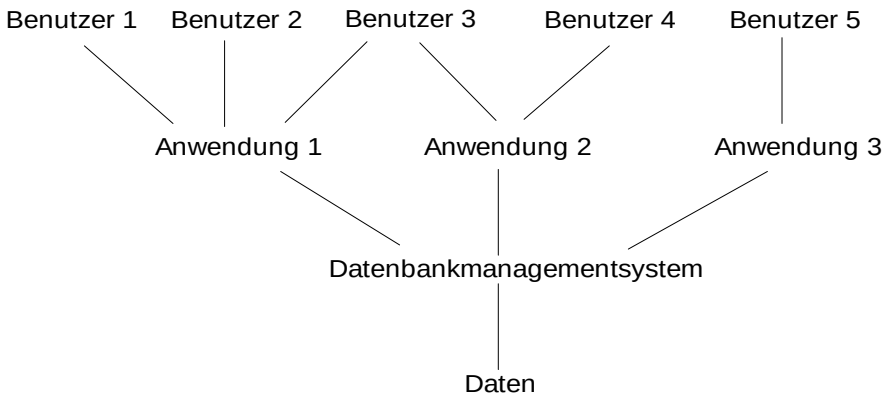


Abbildung 5.1: Ein Datenbanksystem [Rol03]

de Softwareschicht - das Datenbankmanagementsystem (kurz DBMS) abgegeben. Das DBMS fungiert hierbei als Vermittler zwischen Anwendung und Daten (siehe Abbildung 5.1). Die eigentlichen Daten werden weiterhin in Dateien verwaltet. Moderne DBMS bieten jedoch auch die Möglichkeit direkt und ohne Umweg über die Dateischnittstelle des Betriebssystems auf Datenträger zuzugreifen.

Benötigt eine Anwendung nur eine Teilmenge der Daten, so stellt das DBMS verschiedene Sichten (views) auf die Datenmenge bereit. Dies ermöglicht auch die Verschmelzung von Dateien, die eine neue Sicht auf die gesammelten Daten gestattet.

Neben der Bereitstellung von Daten für Anwendungen muss ein DBMS jedoch auch noch andere Anforderungen und Aufgaben erfüllen:

- Kontrolle des nebenläufigen Zugriffs auf Daten bei Mehrbenutzerbetrieb zur Wahrung der Konsistenz der Daten
- Garantie der Sicherheit der Daten (Zugriffsschutz)
- Backup der Daten und Wiederherstellen von Daten nach einem Systemabsturz

Wie diese Anforderungen erfüllt und die interne Speicherung der Daten stattfindet, bleibt der jeweiligen Implementierung eines DBMS überlassen. Eine ausführlichere Beschreibung würde den Rahmen dieser Arbeit sprengen. Hierzu sei auf entsprechende Fachliteratur, wie z. B. [Rol03, EN00], verwiesen. Im

Folgenden sollen nur die zwei wichtigsten Vertreter für DBMS etwas genauer vorgestellt werden.

5.2 Relationale DBMS

Relationale DBMS stellen heute die meistverbreitete Art von Datenbanken dar. Die Idee wurde zuerst in den siebziger Jahren von Dr. E. F. Codd vorgestellt [Cod70, Cod79]. Im relationalen Modell werden die Daten in zweidimensionalen Tabellen dargestellt. Die einzelnen Zeilen einer Tabelle entsprechen grob dem, was man bei normalen Dateien als Eintrag bezeichnen würde und werden als Tupel bezeichnet. Die Attribute eines Eintrags stehen in den Spalten.

Eine oder mehrere Spalten stellen einen Schlüssel dar, der das eindeutige Auffinden von Daten ermöglicht. Über diese Schlüssel können Beziehungen zwischen verschiedenen Tabellen hergestellt werden. Indizes auf bestimmte Spalten ermöglichen es, die Suche in der Tabelle zu beschleunigen. So kann sehr schnell nach bestimmten Werten gesucht werden.

Moderne Systeme enthalten Mechanismen, die eine Sicherung der Daten auch während des Betriebes ermöglichen. Außerdem wird die Konsistenz der Daten auch bei Zugriff mehrerer Benutzer oder bei einem Systemfehler erhalten.

Objektrelationale Datenbanksysteme stellen eine Weiterentwicklung der Relationalen DBMS dar. Hierbei werden objektorientierte Methoden verwendet, um die Datenbank zu modellieren. Außerdem wurden neue Datentypen eingeführt, die es zum Beispiel erlauben, relativ große Objekte in serialisierter Form abzuspeichern.

Nahezu jeder große Hersteller hat seine relationale Datenbank inzwischen um objektorientierte Konzepte erweitert. Der Vorteil dieser Lösung liegt in der Vereinigung der bewährten Technologie mit den neuen Konzepten der Objektorientierung. Leider sind die unterschiedlichen Ansätze nicht kompatibel, so dass man sich auf ein DBMS festlegen muss, wenn man diese benutzen möchte.

Modelliert werden Datenbank-Schemata für relationale Datenbanken in ER-Diagrammen (Entity-Relationship Diagramme) [Che76]. Als Standardsprache

zur Beschreibung von Relationalen Datenbanken und zur Abfrage von Daten aus diesen steht die Structured Query Language (SQL) zur Verfügung. SQL ist ANSI[1]-Standard und hat eine Reihe von Standardisierungsverfahren durchlaufen. Zwar hat jeder Hersteller immer noch spezielle Befehle und Datentypen, aber prinzipiell ermöglicht SQL den beliebigen Austausch von Datenbankmanagementsystemen.

5.3 Objektorientierte DBMS

Relationale DBMS haben aufgrund der Mächtigkeit ihrer Beschreibungsmechanismen und der Standardisierung ihrer Schnittstellen eine marktbeherrschende Stellung unter den Datenbanksystemen eingenommen. Jedoch hat mit der Weiterentwicklung der Computer und auch der Software die Komplexität der zu speichernden Daten zugenommen.

Damit ergeben sich auch Probleme für das relationale Datenbankmodell. Vor allem beim Umgang mit multimedialen Daten und anderen großen Datenmengen, die miteinander in Beziehung stehen, ist der relationale Ansatz überfordert. Zwar lassen sich solche Systeme mit relationalen Methoden umsetzen, das Ergebnis ist jedoch meist keine natürliche oder effiziente Umsetzung der Anforderungen.

Neben den bereits erwähnten objektrelationalen stellen objektorientierte DBMS einen Lösungsansatz dar. Vor allem in den Bereichen Entwurf (CAD) und multimediale Datenbanken können diese Systeme ihre Stärken beweisen. Objektorientierte DBMS (OODBMS) sind Datenbanksysteme, die Objekte speichern und verschiedenen Anwendungen zur Verfügung stellen können. Die Objekte haben dabei nicht nur Eigenschaften, sondern auch ein Verhalten. Sie können über Schnittstellen miteinander kommunizieren und Eigenschaften und Verhalten (Methoden) voneinander erben.

Das Konzept der Objektorientierung ist bereits seit längerem in der Softwaretechnik bekannt und wird von Programmiersprachen wie C++ und Java benutzt. Deshalb gibt es bereits kommerziell verfügbare OODBMS, die direkt Objekte aus diesen Sprachen persistent machen können.

[1] American Naional Standards Institute

All diese Eigenschaften und die Verwendung einer objektorientierten Programmiersprache im Projekt PRINCE führten zu einer genaueren Untersuchung der Verwendbarkeit von OODBMS. Dabei wurden jedoch recht schnell Probleme erkannt, die auch in der Literatur [Rol03] bestätigt werden:

- Zwar existiert ein Standard der ODMG (Object Data Management Group) [Obj04] für OODBMS jedoch wurde dieser zur Zeit der Untersuchung von den wenigsten Herstellern unterstützt. Die meisten Anbieter hatten eigene Interfaces entwickelt, die den Benutzer schon sehr früh im Entwurfsprozess auf eine Datenbank festlegt.
- OODBMS benutzen eine Objekte ID um jedes gespeichertes Objekt eindeutig zu identifizieren. Diese OID wird vom System vergeben. Ein direkter Zugriff auf ein Objekt kann nur über diese OID erfolgen. Sucht man nach Daten, so muss ein navigierender Zugriff verwendet werden. Dabei muss man sich über eine Kette von Zeigern von Objekt zu Objekt „hangeln“, bis die gewünschten Daten gefunden wurden.
- Auf Objekte kann nur über vordefinierte Methoden zugegriffen werden. Soll eine neue Datenmenge erzeugt werden, für die noch keine solche Methode verfügbar ist, so ist dies nicht möglich, auch wenn die Daten prinzipiell in der Datenbank vorhanden sind. Vor allem für die Anwendung in der Mikrosystemtechnik, wo häufig neue Beziehungen zwischen den Daten entdeckt werden, ist dies nicht akzeptabel.

OODBMS besitzen ein sehr großes Potential. Durch die Verwendung des Objektmodells lassen sich beliebige Daten speichern und verwalten. Vor allem mit der zunehmenden Verbreitung von objektorientierten Programmiersprachen werden sie immer interessanter. Sobald eine wohldefinierte einheitliche Theorie und die entsprechenden Standards existieren, stellen sie eine ernstzunehmende Alternative dar [MW00].

5.4 Modellierung

In den bisherigen Abschnitten wurden verschiedene Datenbankmanagementsysteme vorgestellt. Um ein solches System mit Daten zu füllen, muss zu Beginn eine Analyse der zu speichernden Problemdomäne durchgeführt werden.

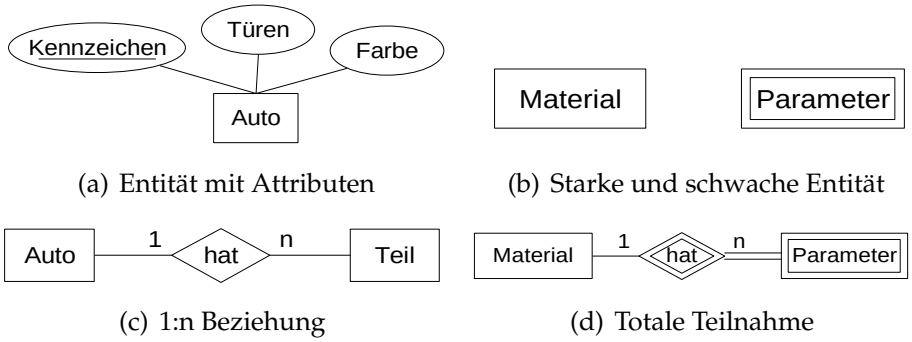
Ziel der Analyse ist es, die Daten aus der realen Welt in Datenobjekte umzuwandeln, die das Datenbankmanagementsystem verwalten kann. Hierzu wird ein Modell auf konzeptioneller Ebene benötigt.

Die Modellierung mit dem Entity Relationship Modell (ER-Modell) wurde bereits 1976 von Chen [Che76] vorgestellt. Seitdem wurde das Modell erheblich verfeinert und ist mittlerweile eines der meistbenutzten Modelle zur Modellierung von Datenbanken. Im folgenden sollen die in Kapitel 6 eingesetzten Methoden zur Beschreibung der Datenhaltung in der Mikrosystemtechnik kurz eingeführt werden.

Wie der Name bereits andeutet verwendet das ER-Modell Entitäten und deren Beziehungen zueinander als Konzepte, um Daten zu modellieren. Entitäten werden benutzt, um Objekte aus der realen Welt darzustellen. Ein Objekt kann hier ein physikalisch greifbares Objekt - wie ein Auto oder eine Person - oder aber auch ein nicht greifbares Objekt - wie ein Bankkonto oder eine Vorlesung - sein. Jede Entität besitzt Attribute, die sie genauer beschreiben. Von besonderer Bedeutung ist hierbei der Primary Key (Schlüsselattribut), der die Entität eindeutig identifiziert. Abbildung 5.2(a) zeigt die Darstellung einer Entität mit ihren Attributen, der Primary Key ist unterstrichen.

Bei den Entitäten kann zwischen starken und schwachen Entitäten unterschieden werden. Starke Entitäten repräsentieren Objekte in der realen Welt, die eigenständig existieren können. Beispiele aus der Mikrosystemtechnik sind das Material oder der Prozessschritt. Mit schwachen Entitäten modelliert man Objekte, die nicht eigenständig existieren können. Sie existieren nur in Zusammenhang mit einer starken Entität. So existieren zum Beispiel Materialparameter nur in Zusammenhang mit einem Material. Schwache Entitäten werden im ER-Diagramm durch eine doppelte Umrandung dargestellt (siehe Abbildung 5.2(b)).

Der zweite Teil im Namen des ER-Modells ist Relationship. Damit wird ausgedrückt, dass die Entitäten in Beziehung miteinander stehen können. Jede Beziehung besitzt Kardinalitäten. Diese geben an, mit wie vielen Objekten jede Entität an der Beziehung teilnimmt. Die Teilnahme einer Entität an einer Beziehung wird durch eine durchgezogene Linie von der Entität zur Beziehung dargestellt. Im Beispiel in Abbildung 5.2(c) wird eine 1:n Beziehung zwischen Auto und Teil hergestellt. Dies bedeutet, dass aus der Entität Auto ein Objekt



(a) Entität mit Attributen (b) Starke und schwache Entität

(c) 1:n Beziehung (d) Totale Teilnahme

Abbildung 5.2: Symbole im ER-Modell

und aus der Entität Teil n - also beliebig viele Objekte verwendet werden. Neben dieser 1:n Beziehung werden noch 1:1 und m:n Beziehungen in dieser Arbeit verwendet.

Alternative Notationen verwenden zum Beispiel eine (min:max) Darstellung, um genauer bestimmen zu können, wie viele Objekte an der Beziehung teilnehmen. Da dies vom J2EE Framework, auf dem die im Projekt PRINCE verwendete Architektur aufbaut (siehe Kapitel 7), nicht unterstützt wird, soll hier nicht näher darauf eingegangen werden.

Auch Beziehungen können Attribute haben. So kann zum Beispiel in einer Beziehung `Auto` *hat* `Rad` ein Attribut **Ort** definiert werden, das angibt, wo am Auto sich das Rad befindet. Die Attribute werden dabei genauso verwendet wie bei Entitäten.

Um die bereits erwähnten schwachen Entitäten an eine starke Entität zu binden, werden die identifizierenden Beziehungen benötigt. Diese werden durch eine doppelte Umrandung im Diagramm dargestellt. Wenn alle Objekte einer Entität an einer Beziehung teilnehmen müssen (z. B. bei schwachen Entitäten) so spricht man von einer totalen Teilnahme (total participation). Diese wird durch eine doppelte Linie von der Beziehung zur Entität dargestellt. Abbildung 5.2(d) zeigt ein Beispiel für Material und Materialparameter.

Das ER-Modell bietet noch viele weitere Möglichkeiten, um Daten auf konzeptioneller Ebene zu beschreiben. Besonders interessant ist, dass es einen Algorithmus gibt, um aus einem ER-Diagramm ein relationales Datenbankschema auf Basis von Tabellen zu erhalten. Dieser Algorithmus wird zum Beispiel in [EN00] und [Rol03] detailliert beschrieben. In diesem Kapitel wurden nur

die im weiteren verwendeten Konstrukte kurz vorgestellt. Eine genauere Beschreibung des Modells kann in der bereits erwähnten Literatur gefunden werden.

5.5 Fazit

Auch wenn objektorientierte Datenbanken die modernere Technologie repräsentieren, und in Kapitel 4 festgestellt wurde, dass viele Analogien zur Objektorientierung vorhanden sind, sind sie wegen der momentan noch mangelnden Standardisierung für unser Projekt nicht geeignet. Die Schnittstellen sind meist von der verwendeten Programmiersprache abhängig, so dass eine Portierung der Daten zu einem späterem Zeitpunkt erschwert wird.

Ein weiterer zu beachtender Punkt ist die Suche in den Daten. Objektorientierte Datenbanken verfolgen einen navigierenden Ansatz für den Zugriff auf die Daten. Für die später umzusetzenden Algorithmen wird jedoch eher ein suchender Zugriff auf Daten nötig sein.

Mit dem ER-Modell wurde ein Modell vorgestellt, das die Modellierung von relationalen Datenbanken vereinfacht. Es ermöglicht die Beschreibung von Problemdomänen auf Basis von Entitäten und ihren Beziehungen. Die Theorie ist ausgereift und es existieren Algorithmen, um die konzeptionellen Modelle in Tabellen eines RDBMS umzusetzen. Aus diesen Gründen fiel die Wahl für die Wissensbasis im PRINCE System auf eine relationales Datenbanksystem. Im folgendem Kapitel werden die Datenstrukturen vorgestellt, die für die Datenhaltung in der fertigungsnahen Mikrosystemtechnik benötigt werden.

6 Datenhaltung in der Mikrosystemtechnik

In Kapitel 2 wurde dargestellt, dass während der Entwicklung neuer Produkte in der Mikrosystemtechnik viele Daten anfallen. Neben den aus der Mikroelektronik bekannten Layoutdaten spielen Prozess-, Material- und Effektdaten eine bedeutende Rolle, da sie über die Funktion und den Aufbau des Produktes entscheiden.

Die untersuchten früheren Ansätze bieten einige Konzepte zur Verwaltung dieser Daten. Wie jedoch in Kapitel 3 dargestellt, sind diese Konzepte nur teilweise geeignet. Um die in Kapitel 4 vorgestellten Entwurfsaufgaben und die verwendete Methodik umzusetzen ist jedoch eine konsistente und permanente Haltung der Daten nötig. Dieses Kapitel stellt die Formalisierung der Daten, die während des PRINCE Projektes gesammelt wurden, vor.

6.1 Objektorientierung in der Datenhaltung

Wie in Kapitel 5 beschrieben, wurde für die Datenhaltung ein relationales DBMS ausgewählt. Im vorgestellten ER-Modell werden die einzelnen Objekte der realen Welt als Entitäten dargestellt, die miteinander in Beziehung stehen. In modernen, objektorientierten Programmiersprache werden jedoch Objekte als Mittel zur Datenhaltung verwendet. Hier findet ein Bruch im Paradigma statt. Um den Einfluss dieser Problematik so gering wie möglich zu halten, wurde eine Systemarchitektur (siehe hierzu auch Kapitel 7) gewählt, die eine Umsetzung der Entitäten in Objekte erleichtert.

Als Objekt in der Datenhaltung wird im Folgendem eine Struktur bezeichnet, die Daten aus einer oder mehreren Entitäten (Tabellen) einem begrifflichem

Objekt aus der Mikrosystemtechnik (z.B. Material, Prozesschritt, etc.) zuordnet.

Die Einführung des Objektbegriffes ist jedoch nur der erste Schritt. Ein Gesichtspunkt der Objektorientierung der sich nur mit zusätzlichem Aufwand umsetzen lässt ist die Vererbung. Prinzipiell gibt es zwar in Datenbankmanagement-Systemen Mechanismen, um Hierarchien darzustellen, diese sind jedoch meist abhängig vom Hersteller der Datenbank. Deshalb sind die im Folgendem vorgestellten Aspekte der Vererbung besser in der Applikationsschicht umzusetzen.

6.1.1 Vererbung

Als erstes stellt sich natürlich die Frage, warum die Einführung eines Vererbungsbegriffes nötig ist. Um dies zu beantworten, müssen die in Kapitel 4 vorgestellten Problemstellungen betrachtet werden.

Als erstes ist hier die Problematik der Klassifizierung zu nennen. In Kapitel 4 wurde verdeutlicht, dass während der frühen Entwurfsphasen nur unklare Spezifikationen für Materialien oder zu verwendende Prozessschritte existieren. Oft möchte man hier nur Klassen angeben (z. B. Material muss ein elektrischer Leiter sein, etc.). Diese Klassifizierung kann als Generalisierung im objektorientierten Umfeld gesehen werden.

Weiterhin wurde im selben Zusammenhang erläutert, dass gleiche Daten (Objekte) unter unterschiedlichen Gesichtspunkten gesehen werden können. So kann zum Beispiel ein Material meist in mehr als eine Klasse eingeordnet werden (z. B. ist Siliziumoxid ein elektrischer Isolator und ein optischer Leiter). Dieser Sachverhalt kann als Mehrfachvererbung interpretiert werden.

Schließlich ergibt sich auch aus der Dateneingabe und -haltung ein Argument für die Vererbung. Durch Vererbung können „Schablonen“ zur Erstellung neuer Datensätze zur Verfügung gestellt werden. Die Arbeit im Umgang mit den Daten wird durch diesen Mechanismus erheblich erleichtert.

Nur durch geeignete Vererbungsstrategien ist der Umgang mit den Daten und auch die Verarbeitung der Daten in der Applikation sinnvoll möglich. Die Anforderungen an die gewählte Vererbungsstrategie unterscheiden sich jedoch zwischen den unterschiedlichen Objekten der Datenhaltung.

Wenn Mehrfachvererbung verwendet wird, so muss sichergestellt werden, dass die Datenstruktur *zyklenfrei* bleibt. Eine Datenstruktur ist zyklenfrei, wenn es beim Durchlaufen der Datenstruktur in eine Richtung (bei Vererbung: Generalisierung oder Spezialisierung) keine Möglichkeit gibt, von einem beliebigen Objekt ausgehend zu dem selben Objekt zu gelangen. Die Zyklenfreiheit ist wichtig, da sonst einige Algorithmen in Endlosschleifen verfallen könnten. Sie muss durch die Applikationsebene sichergestellt werden, da dies mit Mitteln der Datenbank nur schwer möglich ist.

6.1.2 Anpassung der Datenstrukturen

Ein wichtiger Punkt im Kapitel 5 war die Vorstellung des Entwurfs- und Modellierungsmodells - dem ER-Modell. Mit dem ER-Modell wurden relativ strikte Bedingungen, wie eine Datenbank und deren Tabellen zu gestalten sind, eingeführt. Dies wird durch die Normalformen (siehe hierzu [EN00, Rol03]) ausgedrückt. Da jedoch mit einer objektorientierten Sprache und Architektur gearbeitet wird, lassen sich diese nicht immer umsetzen.

Ein Beispiel sind die Primary Keys. In dem in den folgenden Absätzen vorgestelltem Datenmodell werden für die Primary Keys automatisch generierte Objekt-ID's verwendet, auch für die schwachen Entitäten. Diese ID besitzt keinen Informationswert, außer dem, dass sie für jede Entität eindeutig ist.

Die Objekt-ID stellt eine eindeutige ID dar, über die ein Objekt identifiziert wird. Da das Objekt über diese ID auch referenziert wird, kann sie nach der Erstellung nicht mehr geändert werden, ohne die Integrität der Datenbank zu verletzen. Wenn anstelle einer automatisch generierten ID ein realer Wert (wie z.B. ein Name oder eine Formel) stünde, ließe auch dieser sich nach der Erstellung nicht mehr ändern. Wenn also zum Beispiel durch einen Tippfehler das Material nicht Aluminium sondern Alumnium heißt, so kann dieser Fehler nicht einfach korrigiert werden. Um den Fehler zu korrigieren müsste das Objekt und alle Objekte, die dieses Objekt verwenden gelöscht und neu angelegt werden. Bei der starken Vernetzung der einzelnen Entitäten wäre dies sehr fehlerträchtig und zeitaufwendig.

Auch die Verwendung der Vererbung erfordert spezielle Maßnahmen. Um die Vererbung darzustellen, muss die Beziehung zwischen den einzelnen Objekten

in der Vererbungshierarchie gespeichert werden. Da in relationalen Datenbanken keine expliziten Mechanismen zur Darstellung von Vererbung gegeben sind, werden spezielle Felder oder neue Tabellen benötigt.

Im Folgenden werden die einzelnen Objekte der Datenhaltung beschrieben. In den jeweiligen Unterkapiteln wird detailliert auf die durch die Verwendung der Objekorientierung entstehenden Probleme eingegangen. Bei Objekten, die die Mehrfachvererbung verwenden, werden spezielle Strategien zum Umgang mit dieser vorgestellt.

6.2 Daten für alle Objekte

Einige Daten sind für alle Objekte wichtig und sollen deshalb zu Beginn zusammenfassend vorgestellt werden. Diese Daten werden benötigt, um zum Beispiel den Verfasser einer Beschreibung oder das Einstellungsdatum zu erfassen. Sie dienen nicht nur der Dokumentation, sondern können auch für die Aufbereitung der Daten wichtig sein.

Zu diesem Zweck werden für alle wichtigen Entitäten folgende Felder eingeführt:

Autor: ID der Person die die Entität erstellt hat.

EingabeDatum: Datum, an dem die Entität erstellt wurde.

ÄnderungsDatum: Datum, an dem die Entität zuletzt verändert wurde.

ObjectID: Eindeutige ID für die Entität. Sie wird beispielsweise aus der aktuellen Systemzeit, einem Hashwert der Entität und der aktuellen IP Adresse des Rechners errechnet.

Kurzbeschreibung: Eine kurze Beschreibung der Entität, die in der Software als Hilfetext angezeigt werden kann.

Name: Vom Nutzer vergebener Name für die Entität, z. B. Name eines Materials.

In den im Folgendem verwendeten Grafiken wird aus Gründen der Übersichtlichkeit auf die Darstellung dieser Attribute verzichtet.

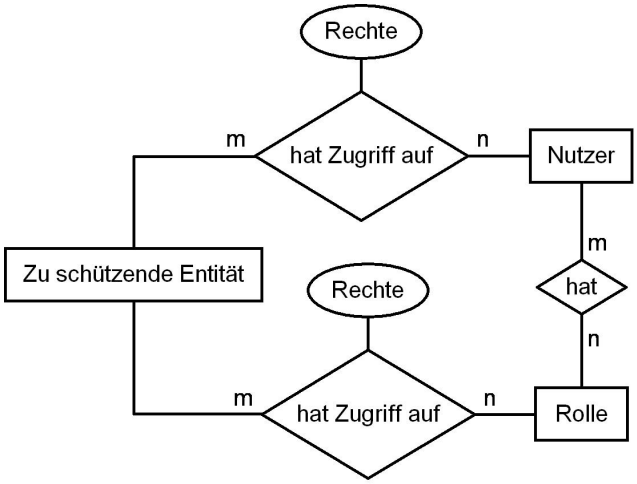


Abbildung 6.1: Verwaltung von Zugriffsrechten

6.3 Nutzerverwaltung

Eine wichtige Aufgabe eines Systemes zur Verwaltung von sensitiven Prozessdaten ist der Schutz vor unauthorisiertem Zugriff. Hierbei müssen die Zugriffsrechte sehr feingranular verwaltet werden. Jeder Datensatz muss sowohl für einzelne Nutzer wie auch für Benutzergruppen freigegeben oder gesperrt werden können. Bei den Zugriffsrechten können hierbei auch noch unterschiedliche Arten von Freigaben (nur lesend, teilweise schreiben, etc.) möglich sein.

Mit der Datenbank selbst ist eine solch feingranulare Regelung nur sehr schwer möglich. Jedoch kann die Datenbank zur Verwaltung von Nutzerlisten dienen. Die auf die Datenbank zugreifende Software regelt anhand dieser Listen, wer bestimmte Datensätze lesen darf oder nicht. Abbildung 6.1 zeigt die Umsetzung solcher Zugriffslisten im PRINCE System.

Umsetzung

Im Zentrum stehen hierbei natürlich die Tabellen für Nutzer und Rollen. Sie enthalten allgemeine Daten, wie Name und Login-Daten des Nutzers beziehungsweise Name und Beschreibung der Rolle. Jeder Nutzer kann mehreren Rollen (z. B. Einzelprozessentwickler und Prozessflussentwickler) angehören. Neben den Nutzer- und Rollendaten werden für jede zugriffsbeschränkte Ta-

belle zwei Tabellen (jeweils eine für Nutzer und eine für Rollen) mit den Zugriffsrechten verwaltet. In Abbildung 6.1 werden diese durch die attributierten Relationen `hat Zugriff auf` dargestellt.

Die Rechte eines Nutzers überschreiben immer die Rechte einer Rolle. So ist es möglich, allen Nutzern, die eine bestimmte Rolle haben, Leserechte auf ein Objekt zu erteilen. Wird einem Nutzer dieses Recht entzogen, das heißt ein entsprechender Eintrag wird in der Rechte-Tabelle für diesen Nutzer gemacht, so können weiterhin alle anderen Nutzer der Rolle auf die Daten zugreifen. Durch diesen Mechanismus kann jedem Datensatz in einer geschützten Tabelle ein individueller Satz an Rechten zugewiesen werden.

Wird ein Nutzer oder eine Rolle aus der Datenbank gelöscht, so werden alle Zugriffsrechte, die der Nutzer oder die Rolle besitzen, auch gelöscht. Eine Rolle kann nur gelöscht werden, wenn keine Nutzer mehr diese Rolle besitzen. Auch wenn ein geschütztes Objekt gelöscht wird, werden aller Zugriffsrechte, die für dieses Objekt existieren, gelöscht. Auf diese Weise wird verhindert, dass die Zugriffslisten zu stark anwachsen.

6.4 Dokumente

Beim Entwurf der Datenhaltung darf nicht vergessen werden, dass am Ende auch Menschen mit den Daten arbeiten sollen. Deshalb muss die Datenhaltung die Möglichkeit bieten, neben den eigentlichen Daten - wie mathematischen Modellen, chemischen Formeln und Parameterdaten - auch Kontextinformationen zu speichern. Zum einen wird es dadurch Neu- und Quereinsteigern erleichtert, sich in die Thematik einzuarbeiten, zum anderen kann dadurch die Glaubwürdigkeit der Daten erhöht werden. In der Mikrosystemtechnik ist dies von besonderer Wichtigkeit, da hier Ingenieure aus verschiedenen Fachrichtungen zusammenarbeiten.

Kontextinformationen können verschiedenartigste Dokumente seien. So reicht es bei einigen Entitäten vielleicht einen kurzen Textkommentar zu speichern, während für andere auf umfassende wissenschaftliche Veröffentlichungen, PDF Dokumente oder Bilder verwiesen werden muss.

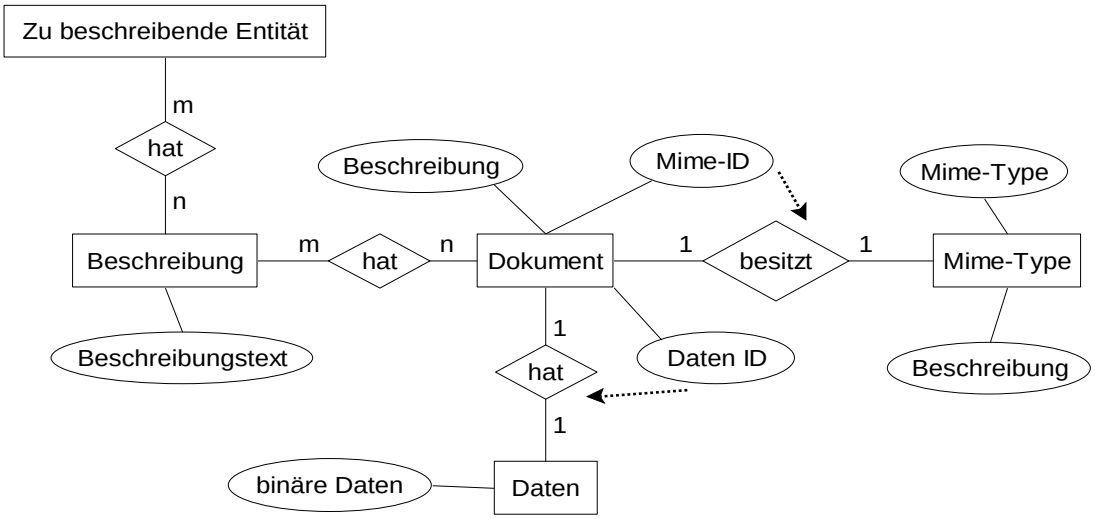


Abbildung 6.2: Formalisierung des Dokumentenmanagements

Document Management System (DMS) und Content Management System (CMS) sind in Zeiten wachsender Informationsflut immer häufiger gebrauchte Begriffe. Im Bereich des Dokumentenmanagements existieren bereits einige verfügbare Lösungen. Als Beispiel sei hier nur das System Documentum [Doc] genannt. Die in dieser Arbeit vorgestellte Lösung soll deshalb nur eine recht eingeschränkte Funktionalität bieten, da später die Verwendung eines professionellen Systems angedacht ist. Für die Grundfunktionalität wurden folgende Kriterien festgelegt:

- Kurze Textbeschreibung zu jeder Entität
- Beliebig viele Dokumente zu jeder Entität
- Klassifizierung der Dokumente nach MIME[1]
- Jede Beschreibung erhält eine eindeutige ID, die einer oder mehreren Entitäten zugewiesen werden kann.

Umsetzung

Anhand dieser Anforderungen wurde das in Abbildung 6.2 vorgestellte Schema entwickelt. Links oben ist der Platzhalter `Zu beschreibende Entität` zu sehen. Dies bedeutet, dass jede Entität der Datenbank, für die Beschreibungen gespeichert werden sollen, über eine m:n Relation mit der Entität `Beschreibung` verbunden ist. Die Entität Beschreibung an sich enthält neben den allgemeinen Daten ein Feld für eine kurze textuelle Beschreibung. Jede zu beschreibende Entität kann mehrere Beschreibungen haben (z. B. für unterschiedliche Sachverhalte oder Untersuchungen) und jede Beschreibung kann mehreren Entitäten zugewiesen werden.

Ähnliches gilt auch für die Beziehung zwischen `Beschreibung` und `Dokument`. Auch hier besteht eine m:n Relation. Die Entität Dokument an sich enthält nur eine kurze Beschreibung des Dokumentes (Metadaten die zur Suche verwendet werden können) und eine aus der Tabelle `Mime-Type` stammenden Mime-Type. Die eigentlichen Daten des Dokumentes werden in einer externen Tabelle gespeichert und nur durch eine ID in der Entität `Dokument` referenziert. Dadurch ist es möglich, die Dateien auch in einem anderen System (z. B. CVS[2]) ohne viel Aufwand zu speichern. Im Feld ID in `Dokument` wird dann lediglich die externe Adresse für das Dokument angegeben.

Die Entität `Mime-Type` enthält die Daten über unterstützte Dateitypen. Neben der eigentlichen Bezeichnung des Mime-Types wird eine kurze Beschreibung des Mime-Types gespeichert. Diese kann auch einen Verweis auf ein Programm enthalten, mit dem entsprechend klassifizierte Dateien geöffnet werden können.

6.5 Parameter und Einheiten

Parameter findet man an vielen Stellen bei der Verwaltung von Fertigungsdaten, seien es nun Prozess- oder Ergebnis-, Material- oder Effektparamter. Deshalb ist es wichtig, sich schon relativ früh Gedanken über die Verwaltung von Parametern zu machen.

[1] Multipurpose Internet Mail Extensions
[2] Concurrent Versions System

Bei der für das Projekt PRINCE durchgeführten Systemanalyse wurde relativ früh festgestellt, dass in den Laborbüchern für den selben Parameter oft verschiedene Namen verwendet werden:

- Druck
- pressure
- Pressure
- ...

Da viele Parameter für spätere Berechnungen, Simulations- und Optimierungsalgorithmen benötigt werden, ist dies in einer firmenweiten Datenhaltung nicht akzeptabel. Deshalb ist eine der wichtigsten Aufgaben der Parameterverwaltung die Sicherstellung der konsistenten Benennung der Parameter.

Ein weiteres Problem ist, dass vor allem im angloamerikanischen Raum viele nicht SI[3]-Einheiten verwendet werden. Deshalb ist die Umrechnung von Einheiten untereinander nicht trivial. Um mit den verschiedenen Einheiten eines Parameters umgehen zu können, muss die Datenverwaltung in der Lage sein, Umrechnungsformeln zu speichern.

Vererbung

Um die Arbeit mit und das Auffinden von Parametern zu erleichtern, ist es nötig eine Klassifizierung durchzuführen. Problematisch hierbei ist, dass je nach Anwendungsgebiet verschiedene Klassifizierungen möglich sind. So kann die elektrische Leitfähigkeit sowohl als Material- als auch als elektrischer Parameter angesehen werden. Da alle diese Klassifizierungen eine Bedeutung für den Nutzer haben können, müssen sie sich auch in der Datenhaltung wiederspiegeln lassen. Deshalb muss es erlaubt sein, Parameter in verschiedene Klassen einzuordnen.

Für den Nutzer sollte diese Datenstruktur völlig transparent sein. Die Grafische Benutzeroberfläche sollte dem Nutzer die aus verschiedenen Programmen gewohnte Baumstruktur anbieten, mit dem Unterschied, dass ein Knoten (Parameter) an mehreren Ästen hängen kann.

[3] Système International d'unités

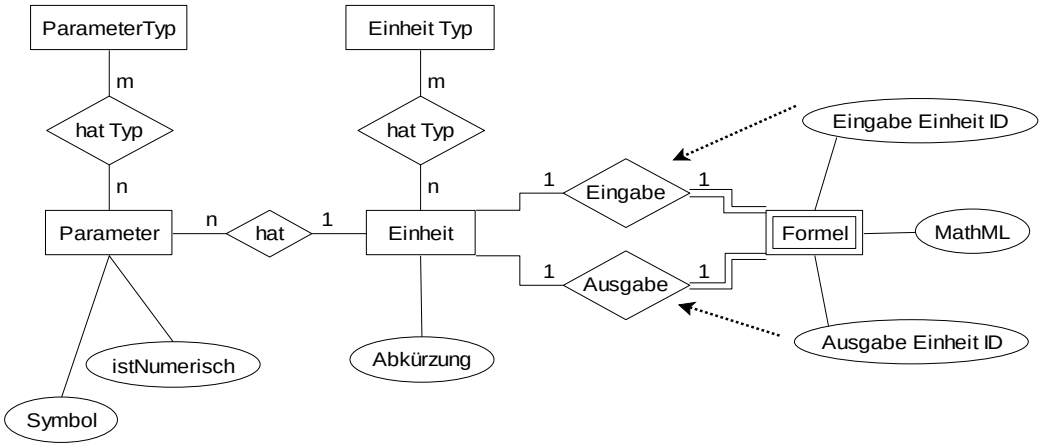


Abbildung 6.3: Datenschema für Parameter

Die Verwendung der Mehrfachvererbung ist in diesem Fall recht einfach möglich. Da die Vererbung nur zur Kategorisierung dient, werden keine Daten im eigentlichem Sinne vererbt. Somit kann es zu keinen Konflikten zwischen den Daten kommen. Im momentanen Schema ist auch nur eine Hierarchieebene vorgesehen, was eine Zyklenfreiheit garantiert. Sollte es sich als nötig erweisen, mehrere Ebenen zu verwenden, so muss sichergestellt werden, dass keine Zyklen in der Hierarchie entstehen.

Umsetzung

Die Abbildung 6.3 zeigt ein Schema der Umsetzung der Anforderung. Links ist die Entität `Parameter` zu sehen. Jeder Parameter kann mehrere Parametertypen haben und jeder Typ wiederum mehrere Parameter, was durch die m:n Relation `hat Typ` dargestellt wird. Die Entität Parameter enthält neben den allgemeinen Daten die Felder `istNumerisch` und `Symbol`. Das erste Feld gibt an, ob die Werte dieses Parameters numerisch (z. B. Druck, Temperatur) sind oder nicht (z. B. Material). Das Feld Symbol enthält die Abkürzung für den Parameter (z. B. U für Gleichspannung).

Jedem Parameter kann nur eine Einheit direkt zugewiesen werden. Dies ist im Normalfall die betreffende SI-Einheit. Jede weitere Einheit wird über die Umrechnungsformeln zugewiesen. Nur wenn für zwei Einheiten Formeln in beide Richtungen vorhanden sind, werden beide dem entsprechenden Para-

meter zugeordnet. Auch für Einheiten wird eine Klassifizierung in Einheitentypen eingeführt. Diese erleichtert die Suche nach bestimmten Einheiten. Jede Einheit kann wiederum mehreren Typen zugeordnet sein.

Jeweils zwei Einheiten können eine Umrechnungsformel erhalten, wobei die eine Einheit die Eingabegröße und die andere die Ausgabegröße darstellt. Die Formeln werden mittels MathML [Wor04b] in der Datenbank gespeichert. In Listing 6.1 wird ein einfaches Beispiel für die Umrechnung von Grad Celsius in Kelvin vorgestellt.

```
<!-- Example: K=C+273.15-->
<Equation>
    <apply>
        <eq/>
        <ci>K</ci>
        <apply>
            <plus/>
                <ci>C</ci>
                <cn>273.15</cn>
        </apply>
    </apply>
</Equation>
```

Listing 6.1: Umrechnung mit MathML

6.6 Materialien

Ein weiterer großer Unterschied zwischen Mikrosystemtechnik und Mikroelektronik ist die Verwendung von einer größeren Vielfalt an Materialien in der Mikrosystemtechnik. Neben den aus der Mikroelektronik bekannten Werkstoffen werden viele Legierungen, Polymere und Metalle verwendet. Allein die Veränderung des Mischungsverhältnisses von Silizium und Germanium kann die Eigenschaften des Materials völlig verändern.

Deshalb ist es nötig, auch Materialien verwalten zu können. Hierbei sollen vor allem die intrinsischen Materialparameter, wie z.B. Schmelzpunkt, Young's

Modul und elektrischer Widerstand, festgehalten werden. Diese Daten können zum einen als Nachschlagewerk und zum Suchen bestimmter, den Anforderungen entsprechender Materialien dienen. Zum anderen können diese Daten auch später Algorithmen zur Verfügung gestellt werden. Auf diese Weise ließe sich zum Beispiel einfach testen, ob ein Prozessschritt die Schmelztemperatur eines vorher verwendeten Materials übersteigt (siehe auch Abschnitt 4.3.2).

Vererbung

Auch bei den Materialien gibt es verschiedene Methoden der Klassifizierung. So kann man zum Beispiel nach Stoffgruppen (Gase, Metalle, Halbleiter, etc.) oder nach Funktion (elektrischer Leiter, Isolator, thermischer Leiter, etc.) unterscheiden. Jede dieser Klassifikationen kann unter bestimmten Gesichtspunkten des Entwurfs eine Bedeutung haben. Des weiteren ist zu berücksichtigen, dass eine Hierarchieebene für Materialien nicht ausreichend ist. Eine Ordnung wie z.B.: Halbleiter - Silizium ist nicht ausreichend, da es von Silizium wiederum einige Unterarten (z.B. porös, amorph, polykristallin, etc.) gibt. Zusätzlich können durch unterschiedliche Prozesseinstellungen auch diese wiederum voneinander abweichende Ausprägungen haben (es gibt mehr als eine Art poröses Silizium).

Diese Art Klassifizierung erfordert den Einsatz der Mehrfachvererbung. Die durch die Mehrfachvererbung entstehende Datenstruktur entspricht einem zyklenfreien Graphen. Die Materialtypen oder -klassen können Parameter beinhalten. Diese besitzen jedoch keine Werte. Alle abgeleiteten Materialklassen und Materialien müssen diese Parameter übernehmen, die Materialien müssen den Parametern zudem Werte zuweisen. Erbt ein Material oder eine Materialklasse von mehreren Klassen, so muss es alle Parameter aller Klassen übernehmen. Tritt ein Parameter mehrfach auf, so wird der entsprechende Parameter natürlich nur einmal übernommen.

Auf diese Weise wird sichergestellt, dass ein Algorithmus der zum Beispiel ein elektrisch leitendes Material sucht, in der entsprechenden Klasse nur Materialien vorfindet, für die auch ein Leitwert vorhanden ist. Durch die Einschränkung, dass Klassen keine Werte haben dürfen, wird der Umgang mit der Mehrfachvererbung vereinfacht. Es können keine leeren Intervalle durch sich nicht überschneidende Wertebereiche entstehen.

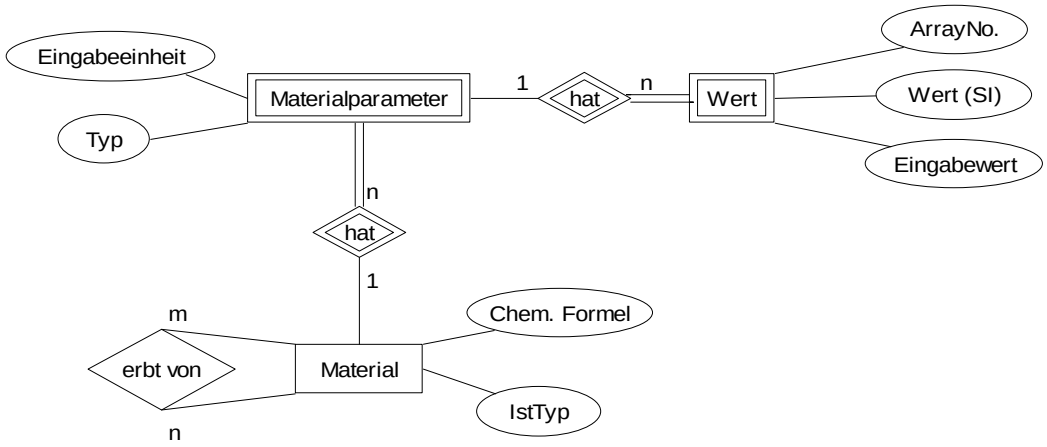


Abbildung 6.4: Datenschema für Materialien

Umsetzung

Abbildung 6.4 zeigt die Umsetzung für die Haltung der Materialdaten. Jeder Datensatz in der Entität `Material` kann ein Materialtyp oder ein echtes Material sein. Dies wird durch das Feld `IstTyp` festgelegt. Eine Hierarchie von Materialtypen und Materialien kann mittels der m:n Relation `erbt von` aufgebaut werden. Jedes Material und jeder Materialtyp kann hierbei von beliebig vielen Materialtypen abgeleitet sein. Von einem „echtem“ Material können keine weiteren Materialien abgeleitet werden.

Jedem Material können Parameter zugeordnet werden. Als Parameter dürfen nur solche gewählt werden, die auch in der Parameterdatenbank spezifiziert wurden. Für jeden Parameter wird eine `Eingabeeinheit` ausgewählt, die die Einheit angibt, in der der Nutzer die Daten eingegeben hat. Mittels des Feldes `Typ` kann festgelegt werden, wie die Werte zu interpretieren sind. Hier kann zum Beispiel angegeben werden, dass die Werte den entsprechenden Richtungen im Kristallgitter zuzuordnen sind. In späteren Versionen ist es durch diesen Mechanismus auch möglich, Intervalle für Materialtypen zu definieren.

Die Entität `Wert` enthält die eigentlichen Werte der Parameter. Durch die Abtrennung ist es möglich, mehrere Werte pro Parameter zu verwalten. Das Feld `Eingabewert` enthält den Wert in Textform, so wie der Nutzer ihn eingegeben hat. Wenn der Parameter numerische Werte hat (durch `istNumerisch` gekennzeichnet - siehe oben), wird zusätzlich im Feld `Wert (SI)` der Wert

als reelle Zahl gespeichert. Falls die Eingabe in einer anderen als der „Haupteinheit“ (siehe Abschnitt Parameter und Einheiten) erfolgt ist, so wird der Wert vorher für diese Haupteinheit umgerechnet.

Der Grund hierfür ist, dass Texte anders als Zahlen sortiert werden. So ist zum Beispiel „524“ größer als „1.000.000“, wenn man die Werte als Text interpretiert. Dies wäre für Suchanfragen (z.B. finde alle Materialien mit einem Schmelzpunkt > 545°C) äußerst hinderlich. Außerdem werden Suchanfragen sehr komplex, wenn mehrere Einheiten für jeden Parameter berücksichtigt werden müssen. Durch die Normierung sind Werte nur noch für eine Einheit zu suchen. Diese Redundanz in der Datenhaltung darf für den Nutzer nicht sichtbar sein und muss auf der Applikationsebene sichergestellt werden.

Das Feld `ArrayNo` ist als Platzhalter zu sehen. Es beinhaltet je nach Typ unterschiedliche Daten. Ist der Parameter zum Beispiel vom Typ Intervall so gibt es in `ArrayNo` die Werte min und max für die untere und die obere Grenze des Intervalls. Ähnliches kann auch für die Kristallrichtung (z. B. durch Miller-Indizes) oder andere Typen von Parametern angegeben werden.

6.7 Prozessschritte

Wie in Kapitel 2 dargestellt, gibt es in der Mikrosystemtechnik eine Vielzahl von verwendeten Methoden zur Fertigung. Bei jeder dieser Methoden stehen wieder unzählige Varianten von konkreten Prozessfolgen zur Verfügung. Bereits kleinste Änderungen an einem einzelnen Prozessschritt können sehr große Auswirkungen auf das endgültige Produkt haben. Außerdem können identische Prozessschritte (wie z.B. RCA-Reinigung) in verschiedenen Prozessfolgen verwendet werden.

Deswegen ist es nötig, die Daten von einzelnen Prozessschritten zu speichern. Unter einem Prozessschritt soll im Folgenden der kleinste, nicht mehr (sinnvoll) unterteilbare Teil einer Herstellungsanweisung für ein mikrotechnisches Produkt verstanden werden.

Für jeden Prozessschritt kann eine große Menge an Daten anfallen. Neben den unmittelbar mit dem Prozessschritt verbundenen Daten, den Prozess- und Ergebnisparametern, müssen auch Daten aus dem Umfeld des Prozessschrittes

gespeichert werden. Es ist außerdem zu erwarten, dass im Laufe der Zeit eine große Menge an Prozessschritten gesammelt werden. Deshalb ist es unerlässlich, sich über eine geeignete Methode der Kategorisierung Gedanken zu machen. In den folgenden Abschnitten werden die einzelnen Gesichtspunkte näher betrachtet.

Vererbung

Einer der wichtigsten mit dem PRINCE-System vorgestellten Neuerungen ist die Einführung eines Vererbungsbegriffes für Prozessschritte. Im Kapitel 4 wurden bereits die Vorteile der Vererbung dargestellt. Neben der Tatsache, dass auf diese Weise einfach Schablonen für Prozessschritte erstellt werden können, wird dieser Mechanismus auch für den Entwurf komplexer Prozessfolgen benötigt. Nur durch die Vererbung ist es möglich, die abstrakten Objekte zu definieren, die für die frühen Entwurfsphasen gebraucht werden.

Die Vererbung bietet jedoch noch weitere Vorteile. So können zum Beispiel Regeln (siehe weiter unten im Text), die für alle Prozessschritte einer Gruppe gelten, einfach durch Vererbung weitergegeben werden. Somit erhält automatisch jeder neue in dieser Gruppe angelegte Prozessschritt diese Regeln.

Das PRINCE-System beschränkt die Datenhaltung jedoch nicht auf eine Einfach-Vererbung[4]. Um das volle Potential eines Entwurfssystems nutzen zu können, musste ein Mechanismus zur Mehrfachvererbung eingeführt werden. Erst die Mehrfachvererbung erlaubt es, Prozessschritte nach unterschiedlichen Gesichtspunkten zu klassifizieren. Ein weiterer Vorteil der Mehrfachvererbung ist, dass Regeln, Kommentare und Parametereingrenzungen von mehreren Typen und auch Maschinen übernommen werden können. Auf diese Weise wird verhindert, dass z.B. Werte ausgewählt werden, die die Maschine nicht zur Verfügung stellt.

Die Mehrfachvererbung bringt jedoch auch Probleme mit sich. Wenn ein Prozessschritt von mehreren Typen erbt, können Widersprüche in den Regeln oder leere Intervalle entstehen. Im PRINCE-System werden deshalb zwei Stufen der Vererbung unterschieden:

[4] Jeder Prozessschritt dürfte dann nur von einem Typen erben

restriktive Vererbung wird für die Regeln angewandt. Jede Regel kann durch einen Nachfolger nur verstärkt, nie aber abgeschwächt werden. Schließen sich zwei Regeln gegenseitig aus, so muss die Regel, die weiter oben im Vererbungsbaum/-graphen angesiedelt ist, abgeschwächt werden. Die Einhaltung dieser Bedingung muss durch die Software[5] überprüft werden, da dies in der Datenhaltung nur schwer realisierbar ist.

nicht-restriktive Vererbung wird für die Parameter verwendet. Für jeden neu angelegten Prozessschritt werden die Grenzen aus den vorgegebenen Intervallen ermittelt. Diese können jedoch überschrieben werden.[6]

Prozessparameter

Prozessparameter sind all diejenigen Größen, auf die der Anwender direkten Einfluss hat. Hierbei werden lediglich die eingestellten und nicht die tatsächlich erreichten Werte betrachtet. Beispiele für Prozessparameter sind:

- der eingestellte Druck
- die eingestellte Temperatur
- die Konzentration einer Lösung
- das verwendete Prozessgas

Das letzte Beispiel zeigt, dass nicht alle gesammelten Werte numerisch sein müssen. Bei den Werten kann es sich auch um Materialien (siehe Kapitel 6.6) oder andere nichtnumerische Größen handeln. Materialien werden hierbei besonders behandelt. In Materialfelder dürfen nur Werte eingetragen werden, die vorher in der Materialdatenbank definiert wurden. So wird eine eindeutige Identifizierung der verwendeten Materialien möglich.

[5] Durchführung einer Revision bei Änderung in der Vererbungshierarchie oder neu eingefügten Regeln

[6] In einer späteren Version der Software ist es vorgesehen, durch das Überschreiben entstehende Konflikte automatisch durch Abschwächung von Parametergrenzen in den entsprechenden Prozessschrittypen aufzulösen. Dies hat jedoch keine Auswirkung auf die Datenhaltung und kann durch die Software umgesetzt werden.

Ein weiteres zu beachtendes Problem ist, dass Parameter im Verlauf des Prozessschrittes unterschiedliche Werte annehmen können. So kann beispielsweise die Temperatur über sogenannte ramp up bzw. ramp down Schritte langsam erhöht bzw. abgesenkt werden. Hierbei sind z.T. wichtige Zeitrestriktionen zu beachten.

Wie in Abschnitt Vererbung erläutert wurde, ist es möglich auch abstrakte Prozessschritte zu definieren. Diese repräsentieren ganze Gruppen von Prozessschritten. Dadurch müssen hier nicht nur diskrete Werte, sondern auch Intervalle, Sequenzen oder eine Auswahl möglicher Werte berücksichtigt werden.

Ergebnisparameter

Ergebnisparameter sind Parameter, die nach oder während der Ausführung eines Prozessschrittes gemessen werden. Beispiele für Ergebnisparameter sind:

- die erreichte Schichtdicke
- das abgeschiedene Material
- der Stress einer Schicht
- die Oberflächenrauheit

Prinzipiell treten bei Ergebnisparametern ähnliche Problemstellungen wie bei den Prozessparametern auf. Jedoch wird durch die Verwendung von Messungen das Problem der Speicherung des Messortes hervorgerufen.

Der Messort beschreibt die Stelle, an der die eigentliche Messung stattgefunden hat. Dies kann beispielsweise der Ort des Temperatursensors in einer Maschine sein. Meistens ist jedoch mit dem Messort der Messpunkt auf dem Wafer gemeint. In Grafik 6.5 wird eine typische Verteilung von Messpunkten (5-Punkt Messung) auf einem Wafer gezeigt.

Vor- und Nachprozessierung

In Kapitel 4.3.2 wurde die Prüfung der Konsistenz einer Prozessfolge als eine der wesentlichen Aufgaben einer die Entwurfsmethodik unterstützenden Soft-

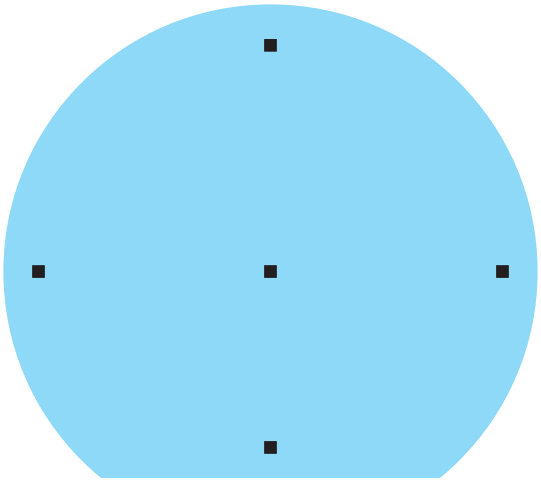


Abbildung 6.5: Messpunkte auf einem Wafer

ware vorgestellt. Um die Prüfung einer Prozessfolge auf Konsistenz durchführen zu können, werden Daten über entsprechende Zusammenhänge benötigt. Hierbei gibt es Regeln, die nicht oder nur indirekt mit dem Prozessschritt zusammenhängen (siehe Abschnitt 6.11) und solche, die direkt mit einem Prozessschritt verbunden werden können.

Für letztere kristallisieren sich bei näheren Untersuchungen des gesammelten Datenmaterials vier Gruppen von Regeln heraus, die berücksichtigt werden müssen. Man unterscheidet Regeln, die:

1. eine Vorprozessierung (z.B. Reinigung) verlangen bzw. verbieten
2. eine Nachprozessierung (z.B. Tempern) verlangen bzw. verbieten
3. die Über-/ Unterschreitung bzw. nicht Einhaltung eines bestimmten Parameters (auch Materialien) verlangen bzw. verbieten, bevor der Prozessschritt durchgeführt werden kann
4. die Über-/ Unterschreitung bzw. nicht Einhaltung eines bestimmten Parameters (auch Materialien) verlangen bzw. verbieten, nachdem der Prozessschritt durchgeführt wurde

Für jede dieser Regeltypen gibt es zwei Stufen, die den Konsistenzcheck beeinflussen:

Recommended: Es wird empfohlen diese Regel einzuhalten, der Prozess kann aber auch gültig sein, wenn dies nicht der Fall ist.

Mandatory: Die Regel muss eingehalten werden, um einen gültigen Prozessfluss zu erhalten.

Mit Hilfe dieser Einordnung können viele Regeln abgebildet werden. Wenn man jedoch komplexere Zusammenhänge betrachtet, wird es nötig, bestimmte Einschränkungen der Gültigkeit dieser Regeln zu verlangen. Hierbei kann wiederum zwischen zwei Einschränkungen unterschieden werden:

zeitliche Einschränkungen: Hier wird bestimmt, ab bzw. bis wann eine Regel gültig ist. Durch das Feld *Immediately* kann bestimmt werden, dass die Regel je nach Typ nur unmittelbar vor bzw. nach dem aktuellen Prozessschritt einzuhalten ist. Dies ist wichtig für bestimmte Prozessschritte, die eine unmittelbare Vor- bzw. Nachbehandlung erfordern (beispielsweise Hotplate direkt nach dem Auftragen eines Fotolackes).

Manchmal kann es auch sinnvoll sein, Regeln durch bestimmte Ereignisse zu aktivieren bzw. zu deaktivieren. Solche Ereignisse können sowohl andere Prozessschritte sein, als auch Parameterwerte, die unter- bzw. überschritten werden. So ist zum Beispiel eine Regel zur Festlegung der Höchsttemperatur durch einen Lackauftrag nur so lange sinnvoll, wie ein Lack aufgetragen ist. Nach dem Prozessschritt zur Entfernung des Lackes (=Ereignis) kann die Regel wieder aufgehoben werden. Durch eine Kombination von Start und Endereignis können Intervalle für die Gültigkeit von Regeln gebildet werden.

räumliche Einschränkungen: Dieser Typ von Einschränkungen ist weit schwerer formal zu fassen. Hierbei handelt es sich um Regeln, die nur auf bestimmte Gebiete angewandt werden sollen. Dies kann zum Beispiel eine Schutzdotierung um einen aktives Gebiet sein. Diese Einschränkungen hängen sehr stark vom Layout und der Intention des Designers ab. Deshalb werden sie in der aktuellen Version der Software nur in informeller Form als Dokumentation (siehe Abschnitt 6.4) gespeichert. Mit der Verfügbarkeit neuer und besserer Daten ist jedoch eine Erweiterung der Datenbasis jederzeit möglich.

Umsetzung

Der Prozessschritt stellt eines der Hauptelemente der Datenhaltung dar. Deshalb ist auch die Umsetzung etwas umfangreicher. Im Mittelpunkt des in Abbildung 6.6 vorgestellten Schemas steht die Entität `Prozessschritt`. Sie beinhaltet alle allgemeinen Daten, wie Autor und Name. Direkt darüber sind die Entitäten `Prozessparameter` und `Ergebnisparameter` angeordnet.

Der Prozessparameter enthält neben den mit dem Materialparameter vorgestellten Daten noch das Attribut `Einheit (Zeit)`. Diese Attribut beinhaltet die Einheit, in der Angaben zur Zeit bei zeitgesteuerten Abläufen eingegeben werden. Die Zeitsteuerung wird genau wie die Angabe eines Intervalls oder einer Auswahl von Werten durch den `Typ` des Parameters definiert. Der Wert der Zeit steht im Feld `ArrayNo`. Bei den Ergebnisparametern wird das Feld `ArrayNo` auch benutzt, um den Messpunkt bei Mehrpunktmessungen anzugeben.

Auch für die Prozessschritte wurde ein Vererbungsbegriff eingeführt. Hierfür wird ein Mechanismus zum Aufbau einer Hierarchie benötigt. In der Datenhaltung wird dies durch eine m:n Relation `erbt von` identisch der Relation bei den Materialien realisiert. Aus Gründen der Übersichtlichkeit wird diese Relation in Abbildung 6.6 nicht gezeigt.

Im rechten Teil der Abbildung befindet sich die Umsetzung einer Datenhaltung für die `Vorprozessierung`. Jeder Prozessschritt kann mehrere „Aktionen“ zur Vorprozessierung besitzen. Dies wird durch die 1:n Relation `hat` realisiert. Eine Aktion kann hierbei die Durchführung eines Prozessschrittes, die Durchführung einer Prozessfolge (siehe Abschnitt 6.10) oder die Einhaltung einer Parameterrestriktion sein. Die drei Realationen `benötigt` stellen diese Beziehung her. Dabei wird für jede Vorprozessierung jeweils nur eine dieser Relationen verwendet. Das Attribut `RegelArt` gibt an, um welche Art (mandatory, recommended, etc.) der Vorprozessierung es sich handelt.

Um die Einschränkung von Aktionen zur Vorprozessierung darzustellen, wurden die Relationen `schränkt ein` eingeführt. Die m:n Relation, die Vorprozessierung mit Prozessschritt verbindet, besitzt das Attribut `Typ`. Der Typ dieser Relation gibt an, ob der entsprechende Pozessschritt oder Prozessschritttyp die Vorprozessierung aktiviert oder deaktiviert und ob er vor oder nach

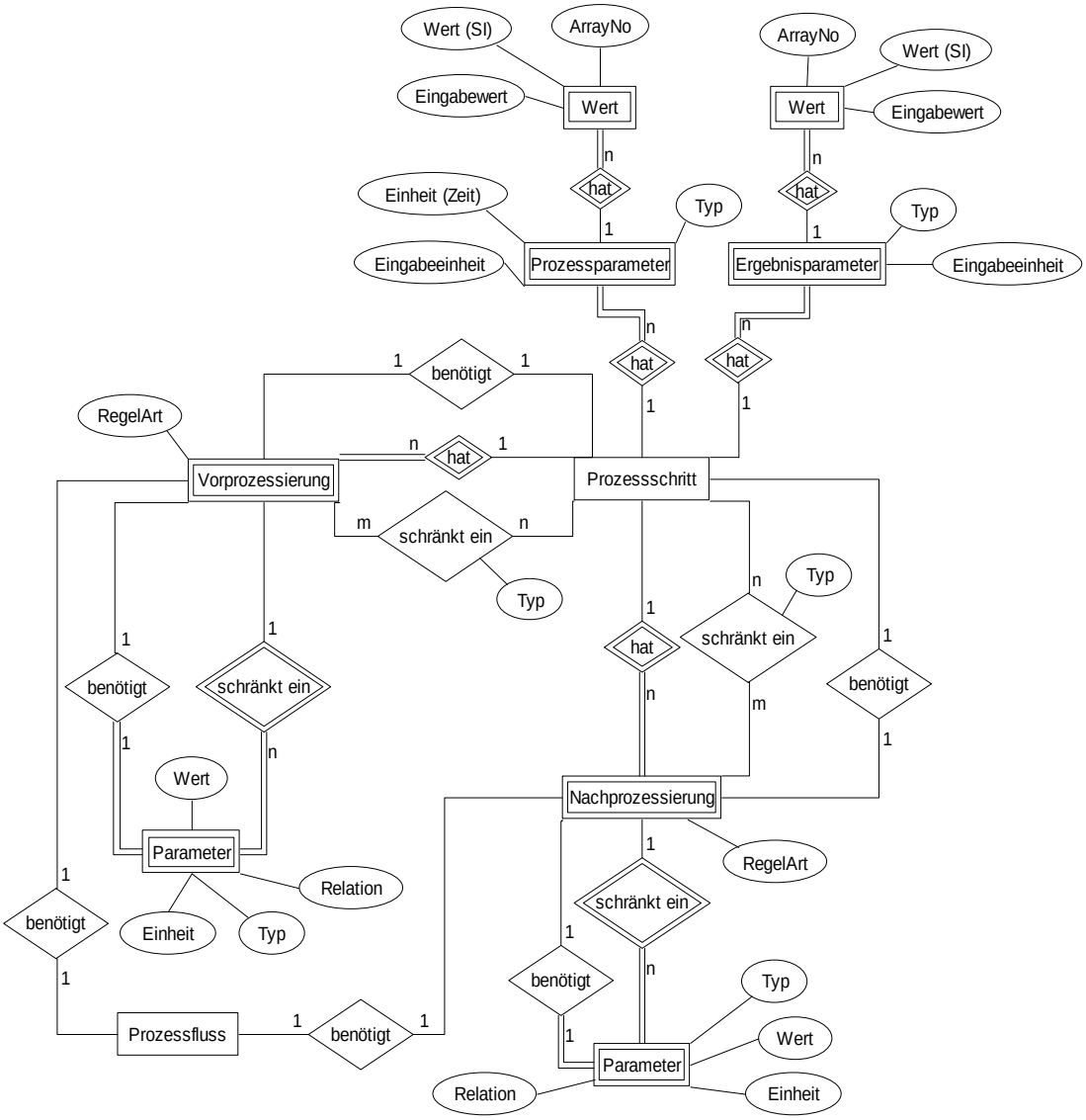


Abbildung 6.6: Datenschema für Prozessschritte

dem aktuellen Prozessschritt gesucht werden soll. Ähnliches gilt für die Einschränkung durch Parameter. Hier befindet sich das Attribut Typ jedoch direkt an der Entität `Parameter`.

Die Entität `Parameter` wird hier zweifach genutzt. Zum einen enthält sie die Parameterwerte, die bei einer Vorprozessierung eingehalten werden müssen. Zum anderen werden hier auch die Einschränkungen durch Parameter gespeichert. Das Attribut `Relation` beinhaltet die Angabe, ob der Wert kleiner, größer oder gleich dem angegebenen Wert sein muss, um der Regel zu entsprechen bzw. um die Aktion einzuschränken.

Im linken unteren Bereich der Abbildung 6.6 befindet sich die Umsetzung einer Datenstruktur für die Nachprozessierung. Der Aufbau ist identisch mit dem der Datenstruktur für die Vorprozessierung.

6.8 Maschinen

Mit Maschinen werden hier allgemein die zur Fertigung benutzten Gerätschaften bezeichnet. Prinzipiell gelten für Maschinen die gleichen Aussagen wie für Prozessschritte. Eine Maschine hat einstellbare Parameter (Prozessparameter) und wirklich erreichte Werte (Ergebnisparameter) und auch für Maschinen können Regeln zur Vor- und Nachbehandlung angegeben werden. Auch die Regeln zur Vererbung sind hier gültig.

Bei Untersuchungen wurde festgestellt, dass auf baugleichen Maschinen die gleiche Fertigungsanweisung zu unterschiedlichen Ergebnissen führen kann. Dies führt dazu, dass die meisten Fertigungsanweisungen nur dann vollständig sind, wenn eine bestimmte Maschnine spezifiziert ist, auf der diese ausgeführt wird. Deshalb werden im PRINCE-System die Maschinen mit den Prozessschritten in einer Datenstruktur gehalten. Alle Fertigungsanweisungen, die auf einer Maschine durchgeführt werden, erben die Daten und Regeln dieser Maschine. In späteren Versionen können beide getrennt werden. Dann müssen jedoch Maßnahmen ergriffen werden, um die Vererbung der Daten und Regeln an Prozessschritte sicherzustellen.

6.9 Wafer

Ein Wafer stellt für die meisten mikrotechnischen Verfahren den Startpunkt dar. Der Wafer kann hierbei aus verschiedenen Materialien, wie z.B. Glas, Kupfer oder dotiertem Silizium bestehen. Da der Wafer nur den Startpunkt einer Fertigungsfolge darstellt, wird er wie ein Prozessschritt behandelt. Es gibt ein „Root-Wafer“, von dem alle Wafer erben. Ein Wafer Objekt kann nur Prozessparameter besitzen, die die Eigenschaften des Wafers beschreiben. Außerdem können für einen Wafer nur Regeln zur Nachbehandlung (z.B. Reinigung) angegeben werden.

Natürlich ist auch interessant, was mit dem Wafer während einer Prozessierung passiert. Diese Daten haben jedoch nicht direkt etwas mit der Fertigungsanweisung zu tun. Deshalb wird die Sammlung dieser Daten in einen Prozess-Tracking Teil der Datenbank verschoben(siehe Abschnitt 6.16).

6.10 Prozesssequenzen

Mit einem Prozessschritt allein lässt sich (in der Regel) noch kein mikrotechnisches Produkt fertigen. Erst die aufeinanderfolgende Ausführung mehrerer Prozessschritte kann zum gewünschten Ergebnis führen. Deshalb muss die Datenhaltung auch diese Folgen (bzw. Sequenzen) berücksichtigen.

Wiederverwendbarkeit

Eine der interessantesten Aspekte bei der Speicherung von Prozessfolgen ist die Wiederverwendbarkeit von Teilen oder Modulen. Es gibt Prozessabläufe, die sich sogar innerhalb einer Prozessfolge wiederholen. Die Arbeit mit Lacken ist hierfür ein Beispiel. Hier trifft man auf die stets nahezu identische Abfolge:

1. Spin On: Auftragen des Lackes
2. Prebake: Härten des Lackes / Austreiben von Lösungsmitteln
3. Lithography: Belichten des Lackes durch eine Photomaske

4. Develop: Entwickeln und Auslösen des belichteten oder unbelichteten Lackes

5. Postbake: Härten des Lackes

Solche Abfolgen können in Modulen gespeichert und dem Nutzer zur Verfügung gestellt werden. Diese Module kann man benutzen, um die Erstellung von neuen Prozesssequenzen zu beschleunigen. Die Software erlaubt eine beliebig tiefe Schachtelung solcher Module (Module können verwendet werden um neue Module zu erstellen). Hierbei muss jedoch darauf geachtet werden, dass keine Selbstrekursion entsteht.

Vererbung

Auch für Prozesssequenzen wurde mit dem PRINCE-System ein Vererbungsbegriff eingeführt. Dieser unterscheidet sich jedoch von dem der Prozessschritte. Im Fall der Prozesssequenzen wird die Vererbung als Mittel genutzt, um Schablonen für neue Prozesssequenzen bereitzustellen.

Erneut liefert der Entwurfsprozess hierfür eine Erklärung. Während sich einzelne Prozessschritte in Kategorien aufteilen lassen, ist dies für Prozesssequenzen nicht so einfach möglich. Jedoch sind die Prozessfolgen häufig ähnlich aufgebaut. Vor allem beim Durchführen von Testreihen werden häufig nur einzelne Prozessschritte verändert.

Deshalb macht es Sinn, Schablonen für solche Prozessfolgen zu definieren. Eine Schablone oder auch abstrakte Prozessfolge kann sowohl abstrakte als auch reale Prozessschritte oder Module enthalten. Um aus einer Vorlage eine reale Prozesssequenz abzuleiten, wird die entsprechende Vorlage gewählt. Die Software sucht automatisch nach abstrakten Prozessschritten in der Folge und gibt passende[7] reale Prozessschritte aus. Der Nutzer kann den richtigen Prozessschritt wählen und so die Folge konkretisieren. Dies wird wiederholt, bis alle abstrakten Prozessschritte ersetzt wurden. Mit abstrakten Prozessmodulen wird auf ähnliche Weise verfahren.

[7] Als passender Prozessschritt gilt jeder, der von dem abstrakten Prozessschritt erbt.

Die Vererbung bei Prozesssequenzen ist nur noch relativ schwach. Alle neu abgeleiteten Folgen können nach Belieben in geänderter Form gespeichert werden und sind nur durch die Vererbungshierarchie an die Schablone gebunden.

Datenzugriff

Bei der Erstellung und Bearbeitung von Prozesssequenzen reicht ein rein navigierender Zugriff auf die Daten nicht mehr aus. Häufig werden Prozessschritte oder Prozessmodule benötigt, von denen nur Parameter, Namen oder Autor bekannt sind. Hierfür wird eine Suchfunktion benötigt.

Die Datenstruktur und die verwendete Architektur (siehe Kapitel 7) wurden mit Augenmerk auf diese Problematik ausgewählt. Mit SQL (siehe Kapitel 5) steht eine mächtige Abfragesprache zur Verfügung. Alle Daten können relativ frei miteinander verknüpft und durchsucht werden. Die in der J2EE Architektur (siehe Kapitel 7) verwendete EJBQL (Enterprise Java Beans Query Language) ist nicht so mächtig wie SQL, reicht jedoch für die meisten Anwendungsfälle aus. Durch eine Kombination beider Techniken kann eine Suchinfrastruktur bereitgestellt werden, die den hohen geforderten Ansprüchen genügt.

Umsetzung

Abbildung 6.7 zeigt schematisch die Umsetzung der Datenhaltung für Prozessfolgen. Die Vererbungshierarchie wird mittels der m:n Relation `ist Template für` realisiert. Da auch hier die Mehrfachvererbung zur Anwendung kommt, muss sichergestellt werden, dass keine Zyklen entstehen. Dies muss in der Softwareschicht umgesetzt werden.

Jede Prozessfolge kann aus Prozessschritten oder anderen Prozessfolgen (Prozessmodulen) bestehen. Dies wird durch die beiden m:n Relationen `besteht aus` ausgedrückt. Das Attribut Sequenznummer gibt dabei die Position des jeweiligen Objektes in der Prozessfolge an. In der Softwareschicht muss sichergestellt werden, dass die Nummerierung korrekt stattfindet.

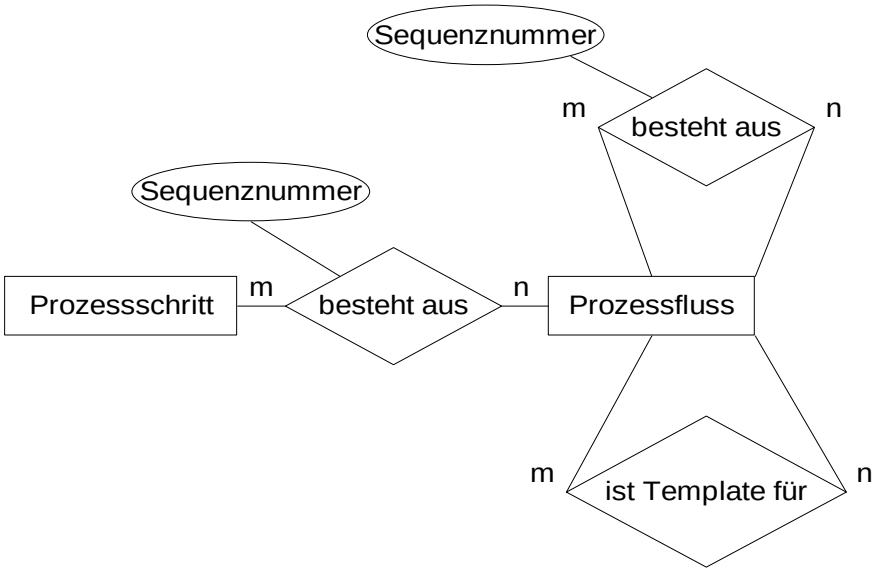


Abbildung 6.7: Datenschema für Prozessfolgen

6.11 Wechselwirkungen

Im Abschnitt Prozessschritte wurden bereits Regeln zur Vor- und Nachbehandlung vorgestellt. Häufig gibt es jedoch wichtige Sachverhalte, die sich über diesem Mechanismus nicht darstellen lassen. Dies ist besonders dann der Fall, wenn andere im Fertigungsgegenstand verwandte Materialien betroffen sind. Bei einem Ätzprozess ist zum Beispiel interessant zu wissen, welche Materialien in welchem Maße angegriffen werden.

Aber auch Wechselwirkungen, die nicht unmittelbar mit dem Prozessschritt in Verbindung stehen, müssen betrachtet werden. Hierbei handelt es sich um Wechselwirkungen zwischen verschiedenen Materialien bzw. zwischen Materialien und Parameterwerten. Beispiele hierfür sind die Haftung zwischen Materialien oder die Änderung der Kristallstruktur durch thermische Einflüsse.

Natürlich ist es wichtig, auch diese Daten zu sammeln. Die gesammelten Daten können später sowohl für eine erweiterte Prüfung der Konsistenz der Fertigungsanweisung als auch für die Übergabe an Simulationstools genutzt werden. In letzteren können sie erheblich zur Steigerung der Genauigkeit der Simulationsergebnisse beitragen.

Effekte, Materialien und Parameter

Wechselwirkungen in der Datenhaltung werden prinzipiell durch einen Effekt (Ätzen, Haftung, etc.) beschrieben. Jeder Effekt kann – muss aber nicht – mit einem Prozessschritt verbunden sein. Auch Materialien (siehe Abschnitt 6.6) können – müssen aber wiederum nicht – an einem Effekt beteiligt sein. Wenn Materialien beteiligt sind, so können sie unterschiedliche Rollen annehmen. So ist zum Beispiel bei einem Nassätzprozess ein Material Ätzmittel, während ein zweites das geätzte Material ist.

Jeder Effekt kann durch Parameter charakterisiert werden. Die Parameter beschreiben den Effekt genauer (z. B. Ätzrate beim Nassätzen). Wenn ein Parameter mehrere Werte hat (z. B. unterschiedliche Ätzraten je nach Gitterstruktur), so muss jeder Wert gespeichert werden können. Dies ist nötig, um die angebundene Simulationssoftware mit Daten versorgen zu können. Natürlich können auch hier nur Parameter und Einheiten verwendet werden, die als solche bekannt gemacht worden sind (siehe Abschnitt 6.5).

Modelle

Mit den bisher vorgestellten Daten ist es möglich die physikalischen Gegebenheiten einer Wechselwirkung zu beschreiben. Um diese Daten sinnvoll nutzen zu können, werden jedoch Modelle benötigt. Ein Modell beschreibt die Weiterverarbeitung bzw. die Aufbereitung der Daten. So kann zum Beispiel ein Ätzeffekt durch ein Gleichungssystem für eine FEM-Simulation oder durch die Zellen eines zellulären Automaten charakterisiert werden [Sch04a]. Dabei können jedem Effekt mehrere Modelle zugeordnet werden. Für jede Aufgabe kann so das beste Modell ausgewählt werden.

Die Untersuchung dieser Modelle ist noch recht neu. In einer Demonstrationsanwendung [Sch04a] wurde das Modell als Speicher für serialisierte Objekte verwendet. In einem Ätzsimulator, der in Form eines zellulären Automaten realisiert wurde, konnte so jedem Material eine entsprechende Zellenstruktur zugeordnet werden. In späteren Projektphasen ist angedacht, die Beschreibung der Modelle mit einer Beschreibungssprache auf Basis von XML durchzuführen. Über entsprechende Schnittstellen sollen diese Beschreibungen an

externe Werkzeuge, wie zum Beispiel Prozesssimulatoren weitergegeben werden.

Vererbung

Auch für die Wechselwirkungen wurde ein Vererbungsbegriff eingeführt. Wechselwirkungstypen haben hierbei mehrere Funktionen. Zum einen dienen Sie der Kategorisierung der Wechselwirkungen. Auf diese Weise kann schnell herausgefunden werden, für welche Simulationstools oder Konsistenzchecks diese Wechselwirkung verwandt werden kann oder muss.

Eine zweite Funktion der Vererbung ist die Bereitstellung von Schablonen für Wechselwirkungen. Hierbei ist die Vererbung restriktiv. In Schablonen haben Parameter keine Werte, jedoch müssen alle in der Schablone vorgegebenen Felder übernommen werden. Die abgeleiteten Wechselwirkungen können um weitere Felder erweitert werden. Um eine Hierarchie von Wechselwirkungen zu ermöglichen, sind mehrere Ebenen von Schablonen möglich.

Eine Mehrfachvererbung ist in der Datenstruktur, nicht jedoch in der Anwendungssoftware vorgesehen. Prinzipiell gelten jedoch die selben Aussagen, die auch für die Materialien gemacht wurden. Falls die Mehrfachvererbung verwendet werden soll, ist lediglich darauf zu achten, dass keine Zyklen in der Hierarchie entstehen. Durch das Verbot von Werten in Typen werden Wertebereichsverletzungen ausgeschlossen.

Umsetzung

In Abbildung 6.8 wird der Aufbau der Datenhaltung für Wechselwirkungen dargestellt. Im Mittelpunkt steht hierbei die Entität `Effekt`. Jeder Effekt wird durch Parameter charakterisiert. Der Aufbau der Parameter für die Entität Effekt ist hierbei identisch mit dem Aufbau der Parameter für Materialien.

Der Effekt besitzt eine Vererbungshierarchie, welche mittels der m:n Relation `erbt von` realisiert wird. Ob eine Effekt ein Typ ist oder nicht, wird mittels des Attributes `istTyp` festgelegt. In der Datenhaltung wird die Mehrfachvererbung unterstützt, obwohl in der bisherigen Anforderungsanalyse kein Be-

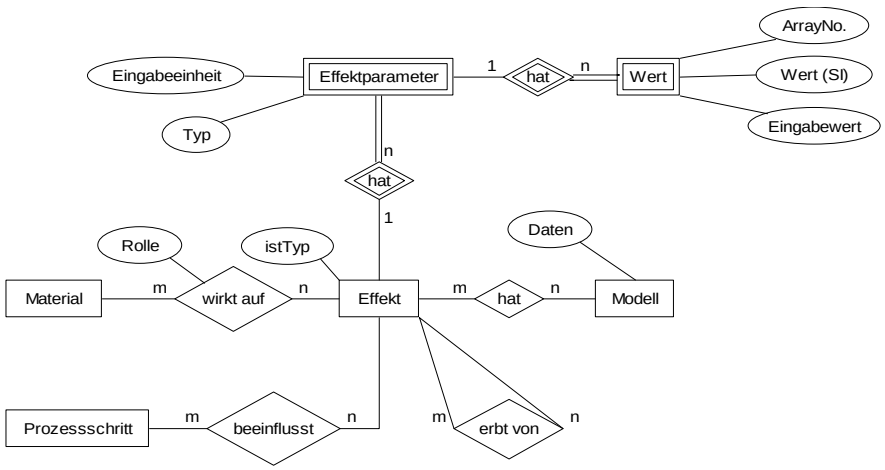


Abbildung 6.8: Datenschema für Wechselwirkungen

darf dafür erkannt wurde. Sollte sich später herausstellen, dass die Mehrfachvererbung doch von Nöten ist, lässt sie sich einfach nachimplementieren.

Die an einem Effekt beteiligten Materialien werden mittels der m:n Relation `wirkt auf` an den Effekt gebunden. Das Attribut `Rolle` an der Relation beschreibt, welche Rolle (z. B. Ätzmittel und geätztes Medium) das entsprechende Material im Effekt einnimmt. Ist ein Effekt auf bestimmte Prozessschritte oder Prozessschritttypen beschränkt, so kann dies mittels der Relation `beeinflusst` dargestellt werden.

Der Begriff des Modells ist, wie bereits erwähnt, momentan noch Gegenstand der Forschung. In der aktuellen Umsetzung besteht ein Modell aus einem Feld für serialisierte Daten. Jedes Modell kann hierbei mehreren Effekten zugeordnet sein und ein Effekt kann mehrere Modelle (z. B. unterschiedliche Genauigkeit des Modells) haben.

6.12 Geometriedaten

Eine Fertigungsanweisung besteht nicht nur aus den Prozessschrittdaten. Um ein Produkt zu fertigen, werden auch geometrische Daten benötigt. In der siliziumbasierten Mikrosystemtechnik werden zur Übertragung der geometrischen Strukturen meist lithographische Masken verwendet. Für diese Masken

wurden spezielle Standardformate entwickelt, die bereits in Kapitel 3.3 vorgestellt wurden. Aber auch für andere Fertigungstechnologien werden geometrische Informationen benötigt. Diese liegen meist in maschinenspezifischen Formaten vor.

Da für diese Daten bereits standardisierte Verfahren zur Speicherung existieren (siehe auch Open Access in Kapitel 3.2.2), werden die Masken ähnlich wie Dokumente behandelt. Für jede Maske werden Metadaten gespeichert, die die Suche erleichtern. Die Daten selbst werden als Objekt serialisiert in die Datenbank abgelegt. In späteren Versionen kann hierfür auch ein Dokumentenmanagementsystem verwendet werden.

Um die geometrischen Daten verwenden zu können, werden sie mit den entsprechenden Prozessschritten (z. B. Lithographie) in den Prozessfolgen in Verbindung gebracht. Ein Datensatz kann hierbei beliebig oft auch innerhalb einer Prozessfolge verwendet werden. Eine Vererbungsstruktur ist für Geometriedaten nicht angedacht. Durch die Metadaten müssen die Geometriedaten hinreichend beschrieben sein, um eine Suche zu ermöglichen.

6.13 Schichten

In Kapitel 4.3.1 wird die automatische Generierung von Prozessfolgen als Entwurfsaufgabe vorgestellt. Hierbei wird eine Schichtfolge als Eingabe verwendet, um eine Prozessfolge zu erzeugen. Die Suche nach geeigneten Schichten und nach Herstellungsverfahren für diese Schichten spielt bei dieser Entwurfsaufgabe eine entscheidende Rolle.

Eine Schicht aus einem bestimmten Material ist meistens das Ergebnis einer Deposition. Wenn dazu noch bestimmte Lithographie- und Ätzschritte verwendet werden, so ist das Ergebnis eine strukturierte Schicht. Jede Schicht hat dabei bestimmte Eigenschaften, wie zum Beispiel den Leitwert oder eine Dicke, die sie charakterisieren. Die Speicherung dieser Parameter ermöglicht eine Suche nach Schichten mit speziellen Eigenschaften. Parameter können hierbei nicht nur diskrete Werte annehmen, sondern auch durch Intervalle oder eine Auswahl möglicher Werte gekennzeichnet sein.

Um die Entwurfsaufgabe der Generierung von Prozessfolgen zu erleichtern, ist es sinnvoll, die Daten der erzeugten Schichten in einem eigenen Datenobjekt zu speichern. Dabei wird jeder Schicht die erzeugende Prozesssequenz zugewiesen. Handelt es sich um eine Strukturierte Schicht, so werden auch die entsprechenden Geometriedaten zugeordnet.

Bei der Suche nach geeigneten Prozessfolgen für die Herstellung eines mikrotechnischen Produktes kann zuerst in dieser Datenstruktur gesucht werden. Werden hier geeignete Schichten gefunden, so stehen bereits Prozessfluss-Module mit allen Hilfsschritten zur Verfügung. Dies beschleunigt den Konsistenzcheck und vermeidet die sonst nötige Suche nach Hilfsschritten zur Erzeugung einer konsistenten Prozessfolge.

Vererbung

Schichten lassen sich nach unterschiedlichen Kriterien klassifizieren. So kann eine Schicht nach Funktion (z. B. elektrischer Leiter), Material (z. B. Metall) oder anderen Eigenschaften spezifiziert werden. Natürlich muss es auch möglich sein, mehrere Stufen in der Hierarchie aufzubauen.

Die Umsetzung dieser Anforderungen erfordert wiederum die Anwendung der Mehrfachvererbung. Für die Parameter wird eine restriktive Vererbung benutzt. Das heißt, dass jeder geerbte Parameter übernommen werden muss. Diese restriktive Vererbung stellt sicher, dass für den Typ wichtige Parameter (wie z. B. der Leitwert für einen elektrischen Leiter) garantiert vorhanden sind.

Die Werte der Parameter können jedoch auch außerhalb der durch die Vererbung gegebenen Intervalle angepasst werden. So wird vermieden, dass Wertebereichsverletzungen zu Fehlern in der Datenstruktur führen. In einer späteren Fassung der Software wird eine automatische Revision der Werte eingeführt. Werte außerhalb der Grenzen, die durch die Typen gegeben sind, führen zu einer Anpassung dieser Grenzen. Dies ist jedoch eine Aufgabe der Implementierung in der Applikationsschicht genau wie die Sicherstellung der Zyklenfreiheit der Datenhierarchie. Die Datenhaltung stellt alle dazu notwendigen Funktionen bereit.

6.14 Komponenten

Auf einer abstrakteren Ebene als die Schichten sind die Komponenten angesiedelt. Ziel der Komponenten ist die Unterstützung des Entwurfes auf hoher Ebene. Eine Komponente stellt ein zu fertigendes Produkt (z. B. Drucksensor) oder einen Teil (z. B. Membran) eines solchen dar.

Jede Komponente wird durch bestimmte Eigenschaften gekennzeichnet. Dies kann zum Beispiel die Genauigkeit des Sensors sein. Diese Werte werden als Parameter zu jeder Komponente gespeichert. Einer realen Komponente (siehe Vererbung) wird außerdem eine Prozessfolge, mit der sie hergestellt werden kann, und einen Satz von Geometriedaten, der die Struktur definiert, zugewiesen. Zusammen stellen Prozessfolge und Geometriedaten eine komplette Fertigungsanweisung für die Komponente dar.

Komponententypen kann eine abstrakte Prozessfolge zugeordnet werden. Dieser Mechanismus erlaubt es, beim Ableiten einer realen Komponente auf einen vorhandenen Prozessfluss zurückzugreifen. Durch Anpassung einiger Parameter (z. B. Schichtdicke) kann so eine neue Prozessfolge für die Komponente auf einfache Weise gefunden werden.

Vererbung

Für Komponenten gelten ähnliche Randbedingungen wie für die Schichten. Sie lassen sich in unterschiedliche Kategorien einteilen (z. B. CMOS-kompatibel und Drucksensor) und auch hier müssen mehrere Ebenen der Klassifizierung möglich sein (z. B. Drucksensor – Drucksensor mit Bereich 1-3bar – Mein Drucksensor).

Zur Umsetzung dieser Anforderungen wird die Mehrfachvererbung verwendet. Die Vererbung ist jedoch für alle Parameter nicht restriktiv. Es werden lediglich Vorschläge unterbreitet, die vom Nutzer überschrieben werden können. Die Zyklenfreiheit muss wiederum durch die Applikationsschicht sichergestellt werden.

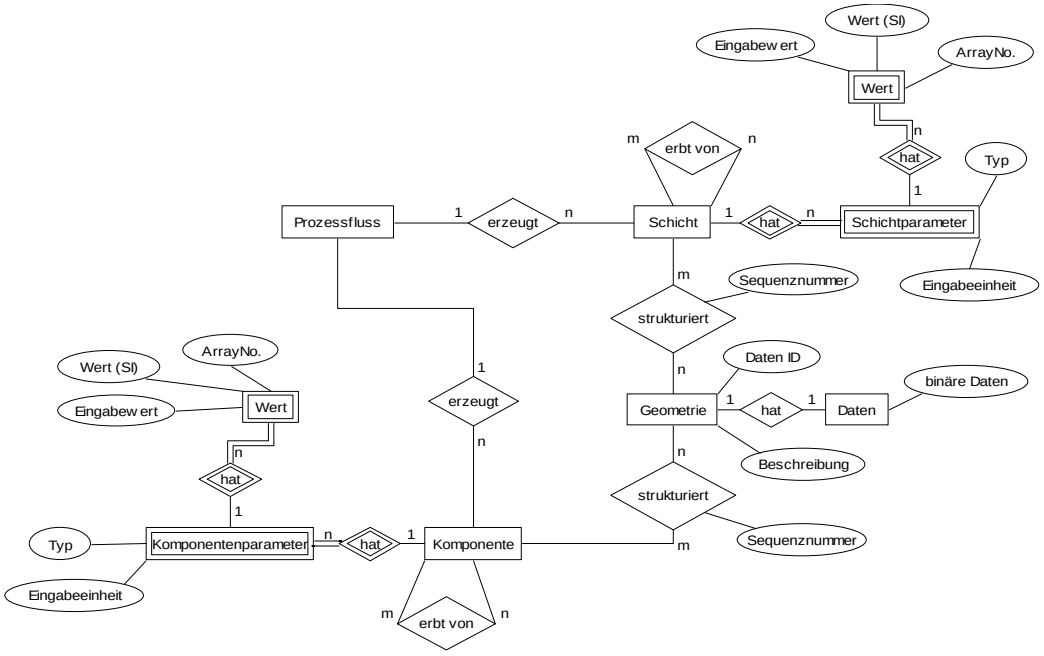


Abbildung 6.9: Geometrie, Komponenten und Schichten

Umsetzung

Abbildung 6.9 zeigt die Umsetzung für Geometriedaten, Schichten und Komponenten. Ähnlich wie bei den Dokumenten werden in der Entität `Geometrie` nur die Metadaten zur Suche in Form der `Beschreibung` und ein Verweis auf den Datensatz gespeichert. So kann die Entität `Daten` jederzeit durch ein anderes Dokumentenmanagement-System ausgetauscht werden, indem der entsprechende Verweis in das Feld `DatenID` der Geometrie geschrieben wird.

Die Datenhaltung für Komponenten und Schichten ist identisch aufgebaut. Für beide Entitäten wird eine Vererbungshierarchie durch die Relation `erbt von` aufgebaut. Hierbei wird die Mehrfachvererbung unterstützt. In der Softwareschicht muss sichergestellt werden, dass in dieser Hierarchie keine Zyklen entstehen. Die Verwaltung der einzelnen Parameter ist wiederum identisch mit der Verwaltung der Materialparameter.

Um eine `Komponente` eindeutig zu beschreiben sind sowohl Geometrie- als auch Prozessdaten nötig. Über die Relation `erzeugt` wird festgelegt, durch welche Prozessfolge die Komponente erstellt wird. Hierbei können durch die

selbe Prozessfolge unterschiedliche Komponenten erzeugt werden (z. B. durch Veränderung der Maskendaten). Über die Relation `strukturiert` werden der Komponente Geometriedaten zugewiesen. Das Attribut `Sequenznummer` enthält dabei die Nummer des Prozessschrittes, auf den die Geometriedaten angewandt werden (z. B. Lithographieschritt). Diese Nummer ist dabei die „absolute“ Position des Prozessschrittes in der Prozessfolge. Diese wird ermittelt, indem alle Prozessmodule aufgelöst werden und so nur eine „flache“ Prozessfolge betrachtet wird.

Für die Schicht wurden identische Relationen angelegt. Eine strukturierte Schicht wird auch durch `Geometrie` und `Prozessfluss` beschrieben. Ist die Schicht nicht strukturiert, wird die Relation `strukturiert` einfach nicht benutzt.

6.15 Das Gesamtschema

Abbildung 6.10 zeigt nochmals in einer vereinfachten Darstellung den Zusammenhang zwischen allen vorgestellten Komponenten der Datenhaltung. Es ist zu erkennen, dass alle Objekte der Datenhaltung sehr stark miteinander in Verbindung stehen. Dabei werden viele der Verknüpfungen nur in der Applikationsschicht realisiert. Die Parameter- und Einheitenverwaltung hat zum Beispiel Einfluss auf alle Entitäten, die Parameter beinhalten. Diese enge Verknüpfung verdeutlicht die Komplexität der Daten, die für einen erfolgreichen Entwurf benötigt werden.

6.16 Partitionierung

Die Datenbank des PRINCE-Systems wurde hauptsächlich als Hilfsmittel zur Entwurfsunterstützung entwickelt. Im Laufe der Arbeiten stellte sich jedoch immer deutlicher heraus, dass zur Unterstützung des Entwurfs eine umfassende Datenbasis benötigt wird. Dies birgt jedoch auch Probleme. Viele der Algorithmen, die für den Konsistenzcheck und die Generierung neuer Prozessfolgen benötigt werden, basieren auf der Suche in der Datenbasis und sukzessiver Einschränkung dieses Suchraumes. Bei einer Vergrößerung des Suchrau-

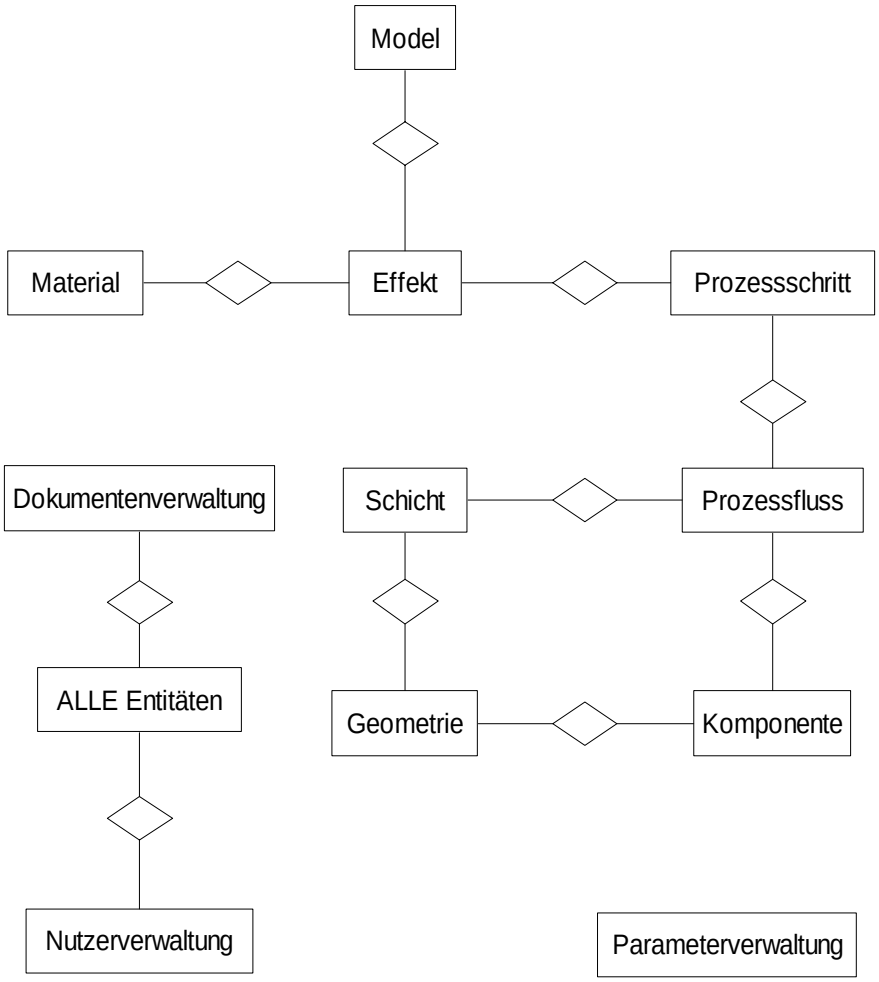


Abbildung 6.10: Vereinfachtes Gesamtschema

mes nimmt auch die benötigte Zeit zur Findung einer Lösung zu. Diese Zunahme wird um so dramatischer, je höherdimensionaler der Suchraum ist.

Auf der anderen Seite ist es aber auch wichtig, so viele Daten wie möglich zur Verfügung zu haben. Mit nahezu jedem Durchlauf eines Rezeptes entstehen neue Daten, die früher oder später an Bedeutung gewinnen könnten. Durch eine systematische Analyse dieser Daten könnten neue Zusammenhänge und Regeln entdeckt werden. Auch sind manchmal Fehlschläge des Einen für einen Anderen sinnvolle und zeitsparende Versuche.

Als Lösung für dieses Problem wird eine Art Partitionierung der Daten vorgeschlagen. Partitionierung heißt in diesem Zusammenhang die Aufteilung der gesammelten Daten. Während eine *Design-Datenbank* weiterhin alle Objekte wie Regeln, Prozessschritte und Folgen enthält, wird eine *Prozess-Tracking Datenbank* aufgebaut, die alle Daten verwaltet, die zur „Laufzeit“ also während der realen Experimente entstehen.

Prinzipiell besteht eine solche Laufzeit-Datenbank aus den Datenobjekten Prozessschritt, Prozessfolge und (evtl.) Schicht. Hierbei können jedoch die Vererbungshierarchie und die Regelbasis vernachlässigt werden. Jede Prozessfolge erbt nur von einer Prozessfolge (Rezept) aus der Designdatenbank. Lediglich Veränderungen in den Prozess- und Ergebnissparametern werden neu aufgenommen.

Jedoch werden für das Prozess-Tracking auch andere Daten benötigt. Hier sind vor allem Daten über den aktuellen Wafer-Status interessant. Diese Problematik wird zur Zeit im Promenade Projekt untersucht. Cavendish Kinetics, einer der Partner in Promenade, hat auf diesem Gebiet bereits Erfahrungen gesammelt. Das von Cavendish Kinetics entwickelte System PDTS (Process Data Tracking System) sammelt Daten über Wafer, Lots und Experimente. Ziel der derzeitigen Forschung ist die Verknüpfung dieser Daten mit dem im PRINCE Projekt entwickelten Datenmodell.

6.17 Fazit

In diesem Kapitel wurde eine erste Modellierung der in der fertigungsnahen Mikrosystemtechnik anfallenden Daten vorgestellt. Es wurde gezeigt, dass

sich alle Daten mit Mitteln eines RDBMS verwalten lassen. Mit der Einführung des Vererbungsbegriffes und der Verwendung von Mehrfachvererbung für ausgewählte Objekte steht ein bedeutendes Mittel für die Umsetzung verschiedener Entwurfsaufgaben zur Verfügung.

Das vorgestellte Konzept ermöglicht eine effektive und strukturierte Verwaltung von Daten. Dem Designer wird die Möglichkeit gegeben, Prozessfolgen unter verschiedenen Gesichtspunkten zu modellieren. In Zusammenhang mit der in den nächsten Kapiteln vorgestellten Software können die benötigten Daten gesammelt werden. Somit stehen die Daten bereit, die Lücke zwischen Verhaltens- und Prozessentwurf zu schließen.

7 Architekturentscheidungen

In den vorigen Kapiteln wurden sowohl die technischen Mittel als auch der strukturelle Aufbau einer effizienten Datenhaltung in der Mikrosystemtechnik vorgestellt. Einer der wichtigsten Fragen bei der Datenhaltung bleibt aber noch zu klären: Wie kommen die Daten in die Datenhaltung? Ein nicht unerheblicher Teil der Zeit während des PRINCE-Projektes wurde für die Suche nach Lösungen auf diese Frage verwendet.

Dabei stellte sich recht schnell heraus, dass die Daten aus den unterschiedlichsten Quellen kommen können. Daten entstehen sowohl an der Maschine wie auch im Büro, zu Hause oder bei einem Partner auf der anderen Seite der Erde. Diese Voraussetzungen stellen gewisse Anforderungen an die verwendete Softwarearchitektur und die zur Erstellung genutzte Programmiersprache. Dieses Kapitel soll einen kleinen Einblick in den stattgefundenen Entscheidungsprozess bieten.

7.1 Anforderungen an die Softwarearchitektur

Wie bereits erwähnt, entstehen Daten an den unterschiedlichsten Orten und werden von verschiedenen Nutzern erzeugt. Dabei möchte jeder Nutzer auf seine oder andere Daten zugreifen können, unabhängig davon, wo er sich gerade befindet. Auch wenn dies aus Sicherheitserwägungen nicht immer möglich sein wird, so sollte der Nutzer zumindestens innerhalb der eigenen Firma unabhängig vom Standort Zugriff auf die Daten haben.

Ein weiteres Problem ist das Vorhandensein verschiedener Hard- und Softwareplattformen. Während die meisten Bürorechner heute PC's mit einer Windows Version sind, werden die Simulations- und Entwurfswerkzeuge meist auf Rechner mit Unix-Derivaten ausgeführt.

Die Datenhaltung selbst liefert schließlich noch eine dritte Anforderung. Fast alle heute verwendeten Programmiersprachen sind objektorientiert. Für die Datenhaltung selbst wird jedoch ein relationales Datenbanksystem verwendet. Die verwendete Softwarearchitektur sollte deshalb nicht nur den Zugriff auf die Daten ermöglichen, sondern auch die Umwandlung der Daten in Objekte erleichtern.

7.2 Plattformunabhängigkeit

Eine der wesentlichen Anforderungen an die zu entwickelnde Software war, dass sie sich einfach auf möglichst viele Soft- und Hardwareplattformen portieren lässt. Während dies auf der rein algorithmischen Seite der meisten heute verwendeten Programmiersprachen möglich ist, entstehen bei der Entwicklung von grafischen Oberflächen ernsthafte Probleme. Die meisten Bibliotheken zur GUI[1] Entwicklung sind abhängig vom verwendeten Betriebssystem. Lediglich zwei ernstzunehmende Alternativen stehen dem Entwickler zur Verfügung:

C++ mit den Qt Bibliotheken von Trolltech [Tro04] Mit Qt steht eine Erweiterung der Programmiersprache C++ zur Verfügung, die ein relativ plattformunabhängiges Programmieren ermöglicht. Dies wird durch die Kapselung der low-level Funktionen des Betriebssystems möglich. Der Programmierer muss nur noch ein Interface für alle gängigen Betriebssysteme kennen. Der Quellcode wird direkt auf den Zielplattformen kompiliert. So entsteht für jede Plattform eine eigene Anwendung.

Sun Java 2 [Sun04b] Die Programmiersprache Java wurde von Sun entwickelt und hat sich zu der objektorientierten Sprache schlechthin entwickelt. Sie zeichnet sich besonders durch ihre Plattformunabhängigkeit aus. Dies wird dadurch erreicht, dass der Java-Code nicht direkt in den plattformabhängigen Maschinencode übersetzt wird, sondern in einen unabhängigen Byte-Code. Dieser wird dann von einem Interpreter zur Laufzeit übersetzt.

[1] Graphical User Interface

7.3 Multi-Tier-Architektur

Die Anforderung des ortsunabhängigen Zugriffs kann nur durch eine Netzwerkarchitektur oder auch Client-Server-Architektur gelöst werden. Client-Server-Architekturen werden gemäß ihres Aufbaus in Modell-Gruppen eingeteilt. Man unterscheidet nach 2-Schicht-, 3-Schicht- und vielschichtigen Modellen. Das 2-Schicht-Modell (engl. 2-tier-architecture) besteht aus einer Anwendungsschicht (engl. application layer) und einer Datenhaltungsschicht. Dieses einfache Modell ermöglicht durch spezifizierte Schnittstellen zur Datenbank, die Datenbank jederzeit ohne großen Aufwand auszutauschen. Allerdings bleibt der eigentliche Programmkern monolithisch.

Im Gegensatz dazu steht das 3-Schicht-Modell (engl. 3-tier-architecture), bei dem ein Teil des Programms auf der Client-Seite läuft, während der andere Teil auf einem entfernten Applikationsserver zu finden ist. Dies gibt dem Entwickler die Möglichkeit, auf der Client-Seite nur die Benutzerschnittstelle, d. h. die Oberfläche zu implementieren. Bei einer solchen Aufteilung der Software spricht man von Thin-Clients. Jegliche Berechnung findet auf dem Server statt, der auch die Kommunikation mit der Datenbank durchführt. Auf diese Weise wird es möglich, viele Änderungen am Code auf dem Server durchzuführen, ohne den Nutzer mit dem Update seiner Client-Software zu belästigen.

Wenn die Berechnungseinheit weiter aufgebrochen wird, so spricht man von einer Multi-Tier Architektur. Dies macht vor allem Sinn, wenn man mit einem Server mehrere Client-Plattformen bedienen muss. Dann kann eine Berechnungseinheit die Daten aufbereiten und Algorithmen auf diese anwenden, während eine andere Einheit die Ergebnisse zum Beispiel für eine Webpräsentation oder einen Client auf Java-Basis aufbereitet.

Ein anderer Grund für das Aufbrechen der Berechnungseinheit ist die Netzwerkbelastung. Viele der in einer Software anfallenden Algorithmen laufen relativ schnell und werden häufig benötigt. Wenn jeder dieser Algorithmen auf dem Server laufen müsste, würde durch die Kommunikation zwischen Client und Server viel Datenverkehr im Netzwerk erzeugt werden. Dadurch würde das Netzwerk sehr stark belastet. Deshalb macht es Sinn, einen Teil der Algorithmen auf der Client-Seite laufen zu lassen.

7.3.1 CORBA

Wenn man mit einer Multi-Tier-Architektur arbeitet, muss man sich natürlich auch mit der Kommunikation zwischen den einzelnen Komponenten - vor allem zwischen Client und Server - Gedanken machen. Hierbei besteht natürlich immer die Möglichkeit, eigene Protokolle zu entwickeln. Um den Aufwand für eine Entwicklung aber so gering wie möglich zu halten, sollten vorhandene Standards verwendet werden.

Die Common Object Request Broker Architecture, kurz CORBA, hat sich in den letzten Jahren zum Quasi-Standard für Client-Server-Architekturen im heterogenen Umfeld entwickelt. Spezifiziert wurde sie von der Object Management Group (OMG). Die Implementierung der Broker-Architektur allerdings wird von ihr nicht durchgeführt. Darum gibt es verschiedene CORBA-Systeme auf dem Markt.

CORBA ermöglicht die Kommunikation von Clients - die in beliebigen Programmiersprachen entwickelt sein können - und Servern, deren Software wiederum in einer anderen Programmiersprache geschrieben worden sein kann. Diese Heterogenität wird durch eine Interface Definition Language (IDL) ermöglicht. Jegliche Schnittstelle, die zur Kommunikation zwischen Client und Server benötigt wird, wird in dieser Sprache spezifiziert. Die Schnittstellen werden dann über entsprechende Compiler in die gewünschte Zielsprache übersetzt. Dabei werden Stellvertreter für den Client auf der Server-Seite (server sceleton) und für den Server auf Client-Seite (client stub) erzeugt. Über diese läuft dann die Kommunikation zwischen Client und Server. Der Programmierer kann die entfernten Objekte wie lokale verwenden.

7.3.2 J2EE

Eine Architektur, die die Kommunikation mit Datenbank und Client standardisiert, wurde mit J2EE [Sun04d] vorgestellt. Die *Java 2 Platform, Enterprise Edition* stellt ein Menge von Spezifikationen und Vorgehensweisen zur Verfügung, die die Entwicklung und das Management von Multi-Tier Anwendungen beschleunigen. Eine allgemeine Darstellung einer J2EE Architektur ist in Abbildung 7.1 zu sehen.

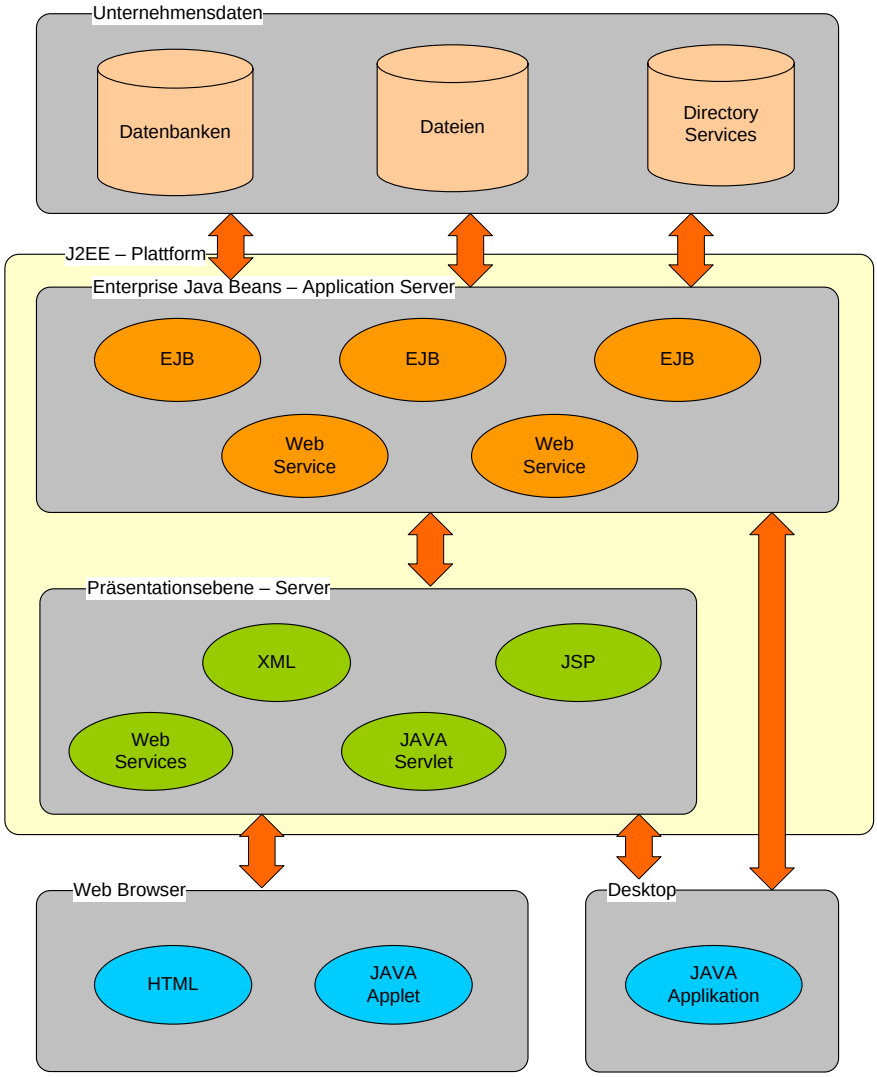


Abbildung 7.1: J2EE Architektur

Der Vorteil der J2EE Architektur ist, dass dem Nutzer jegliche Arbeit den Datenzugriff und die Netzwerkkommunikation betreffend weitgehend abgenommen wird. Der Programmierer kann sich auf die Erstellung der eigentlichen Programmlogik konzentrieren. Die Software auf dem Applikationsserver wird dabei in Komponenten - den so genannten Enterprise Java Beans (EJB) - entwickelt. Jede dieser Komponenten hat eine standardisierte Schnittstelle, die nach Außen bekannt gegeben wird. Dadurch sind diese Komponenten auch in anderen Anwendungen wieder verwendbar.

Für die Kommunikation mit der Datenbank wird JDBC (Java Database Connectivity) [Sun04c] benutzt. Hierbei ist es jedoch möglich, alle direkten SQL Anweisungen durch den Server erstellen und verwalten zu lassen. Der Programmierer arbeitet lediglich mit Objekten, den so genannten *Entity Beans*. Die Programmlogik findet sich in den *Session Beans* wieder, mit denen dann auch der Client kommuniziert. Für spezielle Aufgaben wurden weitere Bean-Typen eingeführt, auf die hier aber nicht näher eingegangen werden soll.

Für die Kommunikation mit dem Client stehen mehrere Möglichkeiten zur Verfügung. Zum einen können über Servlets die Ergebnisse von Anfragen an den Applikationsserver als HTML Seiten übermittelt werden. Als Client dient dann ein Webbrowser, wie zum Beispiel Mozilla oder Opera. Zum Anderen besteht auch die Möglichkeit, mittels vollwertiger Java-Clients Anfragen an den Server zu stellen. Als Übertragungsprotokoll kommt hier RMI (Remote Methode Invocation) [Sun04a] zum Einsatz.

RMI und CORBA sind ähnlich aufgebaut. Der markante Unterschied besteht darin, dass RMI nur für Java entwickelt wurde und wesentlich einfacher zu handhaben ist als CORBA. Es muss keine IDL definiert werden, da nur die Programiersprache Java verwendet wird. Für jede betroffene Bean werden auf dem Server und auf dem Client ein `home` und ein `localhome` Interface definiert, über die die Methoden vom Client aus angesprochen werden können. Ist die Verbindung zur Bean hergestellt, so kann der Client auf alle Methoden des Servers zugreifen, als ob diese lokal im Client implementiert wären. Nahezu alle J2EE Applokationsserver beinhalten mittlerweile auch eine CORBA Implementation, so dass auch mit anderen Programmiersprachen entwickelte Programme auf die Beans zugreifen können.

Mit J2EE steht eine Architektur bereit, die einen schnellen Zugriff auf Daten ermöglicht. Der Umgang mit den Daten wird dabei durch die automatische Umsetzung in Objekte erleichtert. Heute erhältliche Applikationsserver sind schnell und lassen sich bei Bedarf skalieren. Wenn also ein System ausgelastet ist, so können neue Server hinzugefügt werden. Die Verteilung der Last auf die einzelnen Server - das Load Balancing - wird dabei meist automatisch geregelt.

7.4 Client-Software

Auch der Client muss bei den Entscheidungen zur Softwarearchitektur berücksichtigt werden. Insbesondere die Plattformunabhängigkeit spielt eine wichtige Rolle für die Client Software. Die bereits vorgestellte J2EE Architektur bietet hierfür im wesentlichem zwei Möglichkeiten:

Java Client: mittels RMI kann ein in Java geschriebener Client sehr einfach auf die Funktionen des Servers zugreifen. Durch die Verwendung der Java Technologie wird sicher gestellt, dass die Software auf vielen Plattformen zu verwenden ist. Der Java Client kann die gesamte Funktionalität einer Entwurfssoftware bereitstellen und auch mit anderen Tools (z. B. Simulatoren) kommunizieren.

Webclient durch die Aufbereitung der Daten auf der Serverseite kann ein Webinterface bereitgestellt werden. Die so erzeugten Webseiten und Formulare können in jedem gängigen Browser angeschaut und bearbeitet werden. Die Funktionalität einer solchen Oberfläche ist jedoch stark eingeschränkt. Der Webclient eignet sich besonders, um ortsunabhängige Anfragen an und Eingaben in die Datenbank zu erleichtern.

7.4.1 Editoren

Zur Eingabe der Daten in die Datenhaltung werden mehrere Editoren benötigt. Diese Editoren bieten die Möglichkeit, die Daten in der Datenbank zu bearbeiten, zu durchsuchen und verschiedene Prüfverfahren anzuwenden. Im Folgendem werden die einzelnen benötigten Editoren kurz vorgestellt. Aufgrund der

engen Verknüpfung aller Daten müssen die Editoren in der Umsetzung auch eng ineinander verwoben sein.

Das Nutzer-Management bietet alle Möglichkeiten, um neue Nutzer und Rollen anzulegen, bestehende zu bearbeiten und veraltete zu löschen. Dem Nutzer können dabei beliebigen Rollen zugeteilt und auch wieder entzogen werden. Nur ein spezieller Administrator darf die Daten in diesem Editor bearbeiten.

Das Parameter-Management dient zum Verwalten der Parameter und ihrer Einheiten. Die Parameter können in Gruppen eingeordnet und Einheiten können Umrechnungsformeln zugewiesen werden. Jeder Parameter kann direkt nur eine Einheit haben, alle anderen Einheiten ergeben sich aus der Umrechnungsformel. Nur Nutzer mit speziellen Rechten dürfen die Parameterdatenbank ändern.

Der Material-Editor dient zur Bearbeitung der verfügbaren Materialien. Er bietet Möglichkeiten, um Materialgruppen anzulegen und Hierarchien mit diesen aufzubauen. Die einzelnen Materialgruppen und Materialien werden in einer hierarchischen Struktur (z. B. Baum) angezeigt.

Der Effekt-Editor dient der Verwaltung von Effekten zwischen Materialien bzw. zwischen Materialien und Prozessschritten. Jedem Effekt können Materialien mit unterschiedlichen Rollen (z. B. Ätzlösung und geätztes Medium), Prozessschritte und bestimmende Parameter zugewiesen werden. Der Effekt-Editor bietet Möglichkeiten, eine Hierarchie aufzubauen, die in geeigneter Weise (z. B. Baum) dargestellt wird.

Der Prozessschritt-Editor enthält alle Mittel, um Prozessschritte und ihre hierarchische Ordnung zu verwalten.

Der Prozessfluss-Editor dient als Oberfläche für die Erstellung von Prozesssequenzen aus vorhandenen Prozessschritten und Prozessmodulen. Aus dem Prozessfluss-Editor heraus werden auch der Konsistenzcheck und andere Entwurfsaufgaben gestartet.

Hierbei kann der Java-Client alle Editoren enthalten, während für den Webclient nur einige der Editoren mit beschränkter Funktionalität umzusetzen sind.

7.4.2 Verteilung der Software

Trotz der Verwendung einer Multi-Tier-Architektur wird sich natürlich auch der Client im Laufe der Zeit ändern. Da der Client aber mit dem Server kommunizieren muss, ist es wichtig, dass beide auf dem gleichen Stand sind. Deshalb muss ein Weg gefunden werden, wie die Client-Software an die Nutzer verteilt werden kann ohne aufwendige Installationen und hohen Administrationsaufwand.

Für Java-Programme bietet sich hier sofort die Möglichkeit der Java-Applets an. Applets sind Java-Programme, deren Code zentral auf einem Web-Server zur Verfügung gestellt wird. Über spezielle Webseiten ruft der Anwender die Software auf und startet sie in seinem Browser. Der Administrator muss also nur darauf achten, dass auf diesem Server immer die aktuelle Version liegt und hat somit ein gutes Instrument zur Verteilung von Updates. Aufgrund des Sandkasten-Prinzips (engl. Sandbox), das jeder Browser verwendet, ist diese Variante relativ sicher gegen Missbrauch, wie z. B. das Auslesen von Daten von der Festplatte.

Auf der anderen Seite muss der Nutzer jedoch bei jedem Neustart der Software die komplette Anwendung erneut herunterladen, auch wenn sich die Anwendung nicht geändert hat. Zusammen mit den benötigten Bibliotheken zur Kommunikation mit dem Server sind dies jedes mal einige Megabyte an Datenvolumen. Dies gewährleistet zwar eine immer aktuelle Version, verursacht aber auch eine Menge an unnötigen Datentransfer. Ein weiteres Problem ist die Verwendung des Browsers. Dieser benötigt bereits viele Ressourcen, was die Anwendung künstlich verlangsamen würde.

Ein weiterer Weg um Java-Programme zu verwenden, sind die Java-Applikationen. Dies sind eigenständige Programme, die auch außerhalb des Browsers laufen. Deshalb gehen sie schonender mit den Rechnerressourcen um. Problematisch hierbei ist, dass es nicht mehr die Sandbox-Umgebung des Browsers gibt, der den Rechner vor ungeschütztem Datenzugriff bewahrt. Außerdem ändert sich die Applikation - einmal vom Server auf den Rechner geladen - nicht mehr. Bei einem Update muss der Nutzer selbst die neue Version herunterladen und installieren.

Wünschenswert wäre also eine Art der Verteilung, die die Vorteile der Java-Applikation mit denen des Java-Applets verbindet unter Wegfall der Nachtei-

le. Diesen Ansatz verfolgt die Java Web Start Technologie von Sun [Sun04e]. Der große Vorteil von Web Start ist die sehr einfache Nutzung. Bei der Erstellung von Anwendungen muss in keiner Weise Rücksicht auf Web Start genommen werden. Auch schon vorhandene Programme können einfach mittels Web Start zugänglich gemacht werden.

Web Start ist ein Tool, welches sich der Anwender auf seinem Rechner installiert. In aktuellen Versionen der Java Runtime Environment ist Web Start bereits integriert. Da Web Start genauso wie ein Browser nach dem Sandkastenprinzip arbeitet, hat man standardmäßig keinen Zugriff auf Ressourcen außerhalb der Web-Start-Umgebung. Möchte man Ressourcen wie Drucker oder Datei- und Netzwerkzugriff in seiner Applikation nutzen, so muss der Anwender um Erlaubnis gefragt werden. Außerdem werden diese Ressourcen nur Anwendungen zur Verfügung gestellt, die über eine digitale Signatur verfügen. So kann sichergestellt werden, dass nur Code aus vertrauenswürdiger Quelle ausgeführt wird.

Dem Anwender stellt Web Start eine vollständige Java Umgebung zur Verfügung. Auch das Management der Java-Versionen wird von Web Start übernommen. Verlangt eine Anwendung eine nicht vorhandene Version, so wird dies dem Anwender mitgeteilt. Dabei ist es möglich, jeder Anwendung eine andere Java Version zuzuteilen.

Gestartet werden Web Start Applikationen, indem der Anwender auf einer Webseite eine entsprechende Anwendung aufruft. Dabei wird eine Beschreibungsdatei übertragen, die Web Start mitteilt, welche Programmdateien und welche Java Version benötigt wird. Außerdem werden hier auch die Rechte, die die Anwendung benötigt, festgelegt. Danach beginnt Web Start die benötigten Dateien herunterzuladen, legt diese in einem Zwischenspeicher (Cache) ab und startet die Anwendung. Bei einem erneutem Aufruf der Anwendung wird die Anwendung nur heruntergeladen, wenn sich die Dateien auf dem Server geändert haben. Ist dies nicht der Fall, so wird die Anwendung aus dem Cache gestartet. Somit ist sichergestellt, dass die Anwendung immer auf dem neusten Stand ist, ohne das Netzwerk unnötig zu belasten.

Abbildung 7.2 zeigt die Realisierung eines Systems mit Web Start. Die Anwendung wird von einem Webserver heruntergeladen. Nach dem Start kann die Anwendung mit einem Application-Server kommunizieren, der nicht iden-

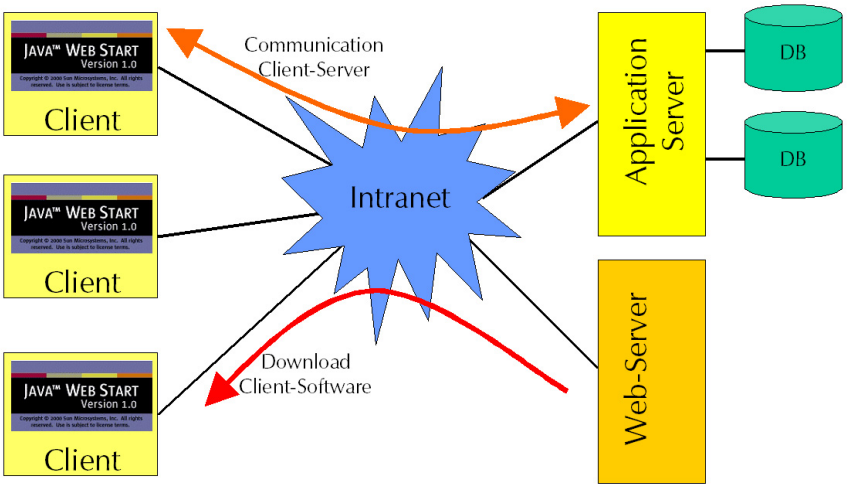


Abbildung 7.2: Web Start

tisch mit dem Webserver sein muss. Aufgrund von Sicherheitserwägungen wird hier nur das Intranet verwendet. Wird die Kommunikation verschlüsselt, könnte auch das Internet zum Einsatz kommen.

7.5 Interfaces

Für einige der in Kapitel 4 vorgestellten Entwurfsaufgaben gibt es bereits sehr weit entwickelte Softwarewerkzeuge, wie zum Beispiel Prozesssimulatoren. In dieser Software stecken oft Jahre an Entwicklungsarbeit. Deshalb wäre es nicht sinnvoll, diese Software neu zu programmieren. Trotzdem sollen diese Entwurfswerkzeuge von der Datenhaltung profitieren. So kann zum Beispiel die Simulation eines Ätzprozesses mit den Daten aus der Materialdatenbank wesentlich genauer durchgeführt werden.

Anforderungen

Um fremde Werkzeuge an die Datenhaltung anzubinden, müssen Schnittstellen definiert werden. Diese Schnittstellen müssen dabei sehr allgemein gehalten sein, so dass möglichst viele Programme auf die Software zugreifen

können. Ideal wäre ein Format, dass unabhängig vom Hersteller der Software ist. Ein solches Format ließe sich dann in sämtliche Herstellerspezifischen Formate umwandeln.

7.5.1 Native Schnittstellen

Mit einer nativen Schnittstelle hat die Software direkten Zugriff auf die Algorithmen des Applikationsservers oder des Clients. Eine solche Schnittstelle kann direkt über RMI oder CORBA implementiert sein. Durch diesen direkten Zugriff ist eine solche Schnittstelle natürlich sehr schnell. Sie erfordert jedoch für jede anzubindende Software erneuten Aufwand, um den Server bzw. den Client entsprechend anzupassen.

Wenn eine solche Schnittstelle als API definiert und allen betroffenen Anwendungen zu eigen wäre, ließe sich der Aufwand zur Anbindung neuer Software erheblich verringern. Mit Open Access (siehe Kapitel 3.2.2) ist dies bereits für Teile des Mikroelektronik-Entwurfes geschehen. Hier können alle Anwendungen, die diese API implementieren, auf eine gemeinsame Datenbasis zugreifen. Leider ist die Standardisierung im fertigungsnahen Bereich noch nicht so weit. Eine entsprechende Definition könnte im Rahmen des Projektes Promenade entstehen.

7.5.2 XML

Als Dateiformat zum Datenaustausch hat sich in den letzten Jahren XML [Wor04a] durchgesetzt. XML, die *eXtensible Markup Language*, kann als Untermenge der eher sperrigen SGML (Standardized Generalized Markup Language - ISO 8879) gesehen werden. Das Ziel von XML ist es, jegliche Daten in strukturierter und maschinenlesbarer Form vorzuhalten.

Der große Vorteil von XML ist die Standardisierung der Syntax. Ein XML Dokument besteht aus reinem Text. Es gibt öffnende und schließende Tags, die die eigentliche Information einschließen. Die Tags ordnen der Information dabei einen Kontext zu oder haben steuernde Funktionen. Listing 7.1 zeigt ein Beispiel für ein XML Dokument.

```
<?xml version="1.0" encoding="UTF-8"?>
<MATERIAL>
    <NAME>
        Silizium
    </NAME>
    <SYMBOL>
        Si
    </SYMBOL>
    <PARAMETER>
        <NAME>
            Dichte
        </NAME>
        <WERT>
            2.3
        </WERT>
    </PARAMETER>
    <PARAMETER>
        ...
    </PARAMETER>
</MATERIAL>
```

Listing 7.1: Beispiel für ein XML Dokument

Der XML Standard an sich ist frei, so dass jeder eine Beschreibungssprache auf Basis von XML entwickeln kann. Es existieren mittlerweile mehrere Parser, die in jedes Programm eingebunden werden können. So ist es relativ einfach, XML Daten in ein Programm einzulesen.

XML an sich beschreibt die Syntax des entsprechenden Dokumentes jedoch nur teilweise. Im Beispiel werden die Tags `DOKUMENT`, `GRUSS` und `NACHRICHT` verwendet. In XML sind alle diese Tags in beliebiger Reihenfolge und Schreibweise erlaubt. Es muss lediglich darauf geachtet werden, dass alle innerhalb einer durch ein Tag geöffneten Umgebung existierenden Umgebungen geschlossen sind, bevor das schließende Tag erscheint. Erst durch eine Spezifizierung einer speziellen Beschreibungssprache wird die Syntax und die Reihenfolge für die Tags festgelegt.

Prominente Vertreter für XML Beschreibungssprachen finden sich heute in vielen Bereichen. So werden zum Beispiel alle WAP-Seiten auf modernen Handys mittels WML (Wireless Markup Language) übertragen. Auch in der Wissenschaft sind Beschreibungssprachen auf Basis von XML in Benutzung. So wurde zum Beispiel MathML als mathematische Beschreibungssprache bereits in Kapitel 6 benutzt, um Umrechnungsformeln zu beschreiben.

PDML

Mit PDML (Process Description Markup Language) [Kle02, WPHB02a] wurde die Umsetzung einer Beschreibungssprache für den fertigungsnahen Mikrosystementwurf vorgestellt. Ziel dieser Arbeiten war es, die Daten aus dem PRINCE-System exportieren und in das System importieren zu können. Hierbei wurden hauptsächlich die Elemente wie Prozessschritt und Prozessfluss berücksichtigt. Eine genaue Beschreibung des aktuellen Standes von PDML ist in [Kle02] zu finden.

Im Rahmen des EU-Projektes Promenade wird eine Weiter- oder Neuentwicklung von PDML angestrebt, die auch die neuen Aspekte, wie die Anbindung verschiedener Simulatoren einbezieht. Besonders Strukturdaten werden hier eine besondere Bedeutung gewinnen. PDML soll als unabhängiges Austauschformat mit anderen Anwendungen dienen.

7.6 Gewählte Architektur

Abbildung 7.3 zeigt eine Darstellung der im PRINCE Projekt umgesetzten Architektur. Als Basis dient eine relationale Datenbank[2]. Sie dient zur persistenten Haltung der Daten.

Als Middle-Tier wurde ein J2EE Applikationsserver[3] eingesetzt. Hier werden die Daten aus der beziehungsweise für die Datenbank aufbereitet. Alle datenintensiven Algorithmen, wie z. B. der Konsistenzcheck, laufen hier. Der

[2] Im Projekt wurde eine Datenbank von Oracle [Ora04] eingesetzt. Prinzipiell ist aber jede SQL-konforme Datenbank einsetzbar.

[3] Im Projekt wurde der OC4J (Oracle Containers for Java) verwendet. Prinzipiell kann jeder EJB2.0 konforme Server verwendet werden.

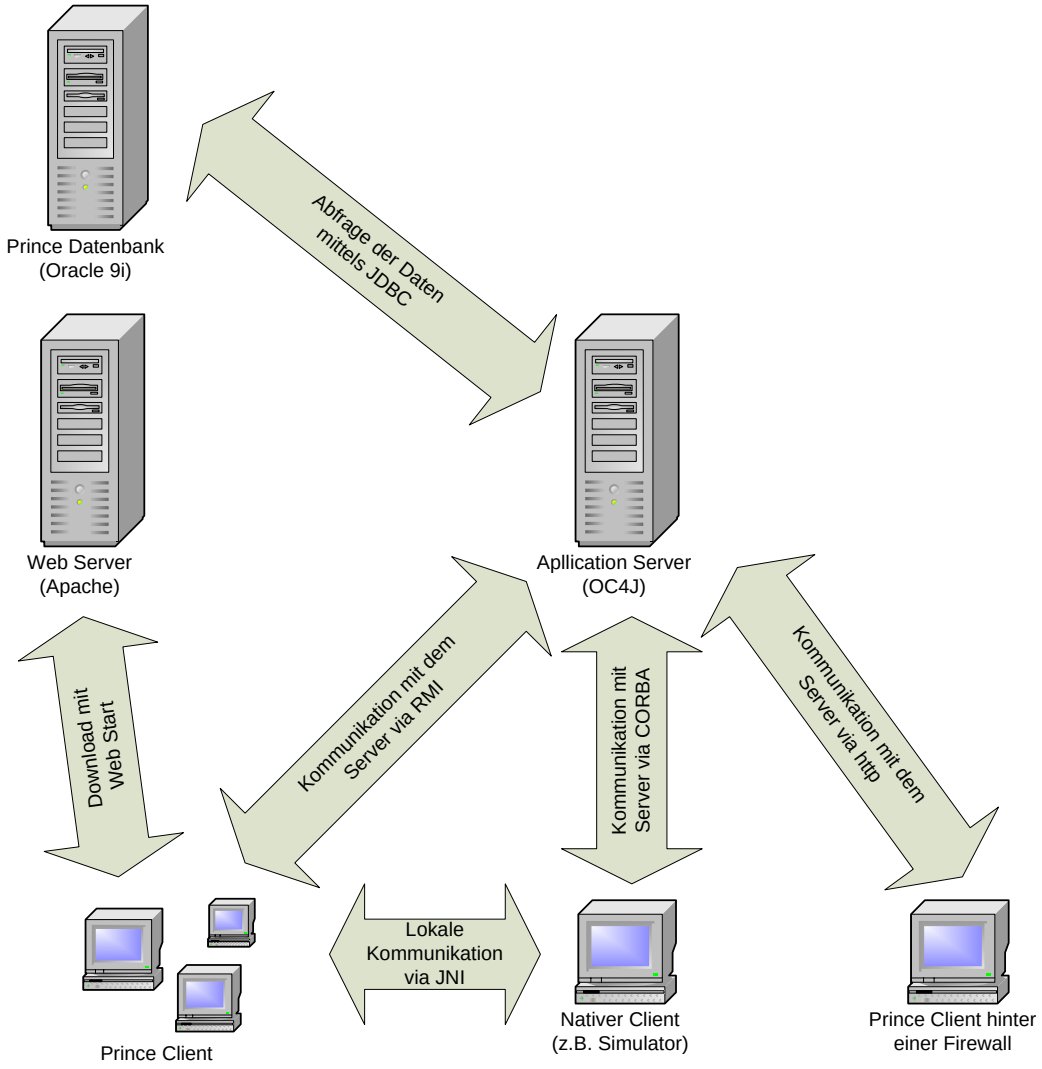


Abbildung 7.3: Schematische Darstellung der Architektur

Applikationsserver ist auch teilweise für die Prüfung der Datenkonsistenz zuständig, indem er die Objekthierarchien auf Zyklenfreiheit überprüft.

Als Client wurde während des Projektes ein Java-Client entwickelt, der die in diesem Kapitel vorgestellten Editoren enthält. Er kommuniziert mittels RMI mit dem Applikationsserver. Da dieses Protokoll durch viele Firewalls ausgefiltert wird, wurde auch eine Übertragung der Daten mittels des http-Protokolls erprobt. Dabei werden die RMI Daten in http-Anfragen „verpackt", so dass sie durch die Firewall gelangen können. Diese Methode hat aufgrund der mehrfachen Wandlung des Protokolls jedoch den Nachteil, dass sie nur eine sehr langsame Kommunikation ermöglicht.

Der Java-Client kann mittels JNI (Java Native Interface) andere native Anwendungen (z.B. Prozesssimulatoren) starten. In späteren Versionen ist vorgesehen, die Kommunikation mit anderen Anwendungen über eine allgemeine XML Sprache wie PDML zu realisieren. Ein entsprechender Client zum Im- und Export von PDML Daten wurde realisiert [Kle02]. Für einige spezielle Anwendungen wurde auch ein direkter Zugriff auf den Applikationsserver mittels CORBA erprobt.

7.7 Fazit

In diesem Kapitel wurde eine Architektur vorgestellt, die die Datenhaltung nicht nur ermöglicht, sondern auch aktiv unterstützt. Sie ermöglicht die Anbindung verschiedener Editoren, mit denen die Daten bequem eingegeben und visualisiert werden können. Offene Standards für Interfaces zu dieser Architektur - und damit zu den realisierten Algorithmen - ermöglichen die Anbindung anderer Software.

Im folgenden Kapitel wird eine erste prototypische Realisierung einer Software vorgestellt, die sowohl die in Kapitel 6 vorgestellte Datenstruktur als auch die in diesem Kapitel vorgestellte Architektur umsetzt.

8 PRINCE

Auch in der Industrie wurde erkannt, dass die fertigungsnahen Aspekte des Mikrosystementwurfs im zunehmenden Maße die Konkurrenzfähigkeit von Produkten bestimmt. Deshalb nahmen bereits im Jahr 2001 Mitarbeiter der Robert Bosch GmbH Kontakt mir unserem Institut auf, um die Problematik eingehender untersuchen zu lassen. Als Resultat entstand das bereits mehrfach erwähnte PRINCE-System. PRINCE ist eine Abkürzung für **PR**ocess **IN**formation and management **CE**nter. In dieser Software wurden die in dieser Arbeit und in [Wag05] vorgestellten Konzepte umgesetzt. Dabei flossen unter anderen auch wertvolle Hinweise der Mitarbeiter der Robert Bosch GmbH in Gerlingen und der Mitarbeiter des Instituts für Halbleiterelektronik (IHE - heute Teil des Instituts für Mikrosystemtechnik der Universität Siegen) in die Produktentwicklung ein.

Wie bereits in Kapitel 7 beschrieben, wurde für die Umsetzung eine Mehrschichtarchitektur verwendet. Die Datenbankebene wurde bereits ausführlich in Kapitel 6 vorgestellt. In diesem Kapitel sollen kurz die Komponenten auf dem Server (Beans) und die Client-Software vorgestellt werden. Eine ausführliche Beschreibung der Funktionalität der Client-Software ist im Benutzerhandbuch [Ins04b] zu finden.

8.1 Server Beans

Auf der Serverseite werden Java Beans verwendet. Hier unterscheidet man im wesentlichen zwischen Entity Beans, die Datenobjekte repräsentieren und Session Beans, die die Logik enthalten. Die Session Beans greifen auf die Entity Beans zu, um an die Daten zu gelangen. In den Session Beans werden diese Daten dann verarbeitet und für den Client aufbereitet. Für die einzelnen Aufgabenbereiche wurden folgende Session Beans entwickelt:

UserManager: enthält alle Routinen um Nutzer und Gruppen anlegen, bearbeiten und löschen zu können. Außerdem bietet der UserManager die Möglichkeit, das Passwort eines Nutzers zu prüfen. Erst wenn der Nutzer sich erfolgreich eingeloggt hat, kann er auf die anderen Beans zugreifen.

ParameterUnitHandler: verwaltet die Daten der Parameter und Einheiten. Hier können neue Parameter, Einheiten und Umrechnungsformeln für Einheiten in die Datenhaltung eingeführt und verwaltet werden.

DocumentHandler: verwaltet Dokumente und Beschreibungen zu den einzelnen Datenobjekten.

ProcessStepHandler: enthält die Methoden, um Prozessschritte suchen, anlegen, bearbeiten und löschen zu können. Hier werden auch die Regeln zur Vor- und Nachprozessierung verwaltet. Der ProcessStepHandler bietet Methoden, um die Vererbung zu realisieren und geerbte Regeln und Parameter aus der Datenhaltung zu extrahieren.

ProcessSequenceHandler: dient zur Verwaltung der Prozessfolgen. Der ProcessSequenceHandler bietet Methoden, um die Prozessfolge aus ihren einzelnen Teilen zusammenzusetzen und übergebene Folgen wieder in den einzelnen Teilen zu speichern. Hierzu werden auch Methoden des ProcessStepHandler genutzt.

EffectHandler: hier werden die Effektparameter, Modelle, beteiligten Materialien und deren Rollen im Effekt verwaltet.

MaterialHandler: bietet alle Methoden, um Materialien und die Materialhierarchie zu verwalten.

SimulationHandler: erhält als Eingabe eine Startstruktur, eine Prozessfolge und Maskendaten und liefert als Ergebnis eine mit den Daten der Prozessfolge und der Masken simulierte Struktur (siehe Abschnitt 8.3.2)

ConsistencyChecker: enthält alle Methoden, die eine Prozessfolge auf Konsistenz prüfen. Eine genauere Beschreibung der Funktion wird in Abschnitt 8.3.1 gegeben.

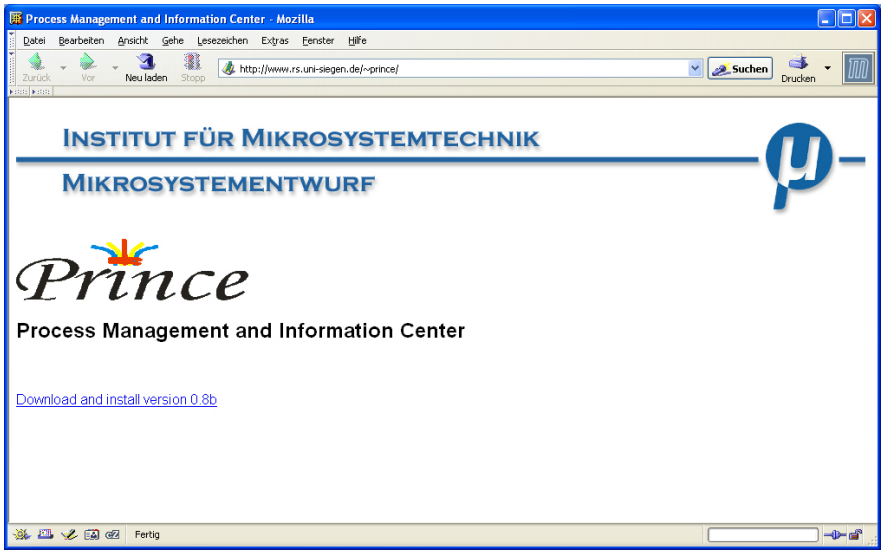


Abbildung 8.1: Webseite zum Start von PRINCE

8.2 Der Prozessschritteditor

Die Verwendung von Web Start zur Verteilung der Client-Software ermöglicht die Bereitstellung im Internet. Zu diesem Zweck wurde eine Webseite eingerichtet, von der aus die Software installiert werden kann (siehe Abbildung 8.1).

Nach dem Download startet die Software. Es erscheint der Login Dialog (Abbildung 8.2). Hier muss sich der Nutzer identifizieren. Außerdem kann der Server und das Kommunikationsprotokoll gewählt werden. Befindet sich der Nutzer hinter einer Firewall, so kann durch HTTP-Tunneling trotzdem eine Verbindung zum Server hergestellt werden. Diese Kompatibilität wird jedoch durch Geschwindigkeitseinbußen erkauft.

Durch das Login wird der Nutzer authentifiziert. Dies erlaubt dem System festzulegen, welche Rechte der Nutzer hat. Die Rechte beziehen sich nicht nur auf den Zugriff auf einzelne Prozessschritte sondern auch auf die Verwaltung von Parametern und Nutzern. Nach erfolgreichem Login erscheint der Prozessschritt-Editor (Abbildung 8.3). Durch die Menüs in diesem Editor können alle anderen Editoren des Systems aufgerufen werden.

Der Prozessschritt Editor dient der Verwaltung einzelner Prozessschritte. Die Prozessschritte werden dabei in eine aus dem Explorer bekannten Baumstruktur eingeordnet. Prozessschritt-Typen werden durch ein rotes Symbol gekenn-

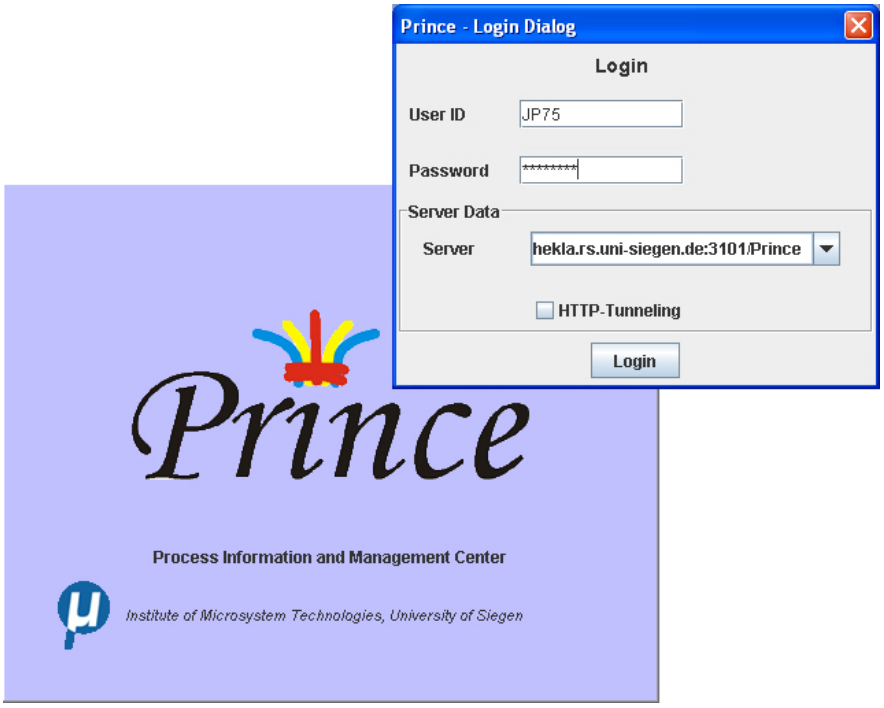


Abbildung 8.2: Login-Fenster von PRINCE

zeichnet während „reale“ Prozessschritte (oder auch Rezepte) ein mehrfarbiges Symbol besitzen. Die Mehrfachvererbung ist für den Nutzer transparent in die Software integriert. So erscheint der Prozessschritt **O2 Flash** sowohl als Reinigungs- als auch als Ätzschritt. Eingegeben werden die Schritte, von denen ein Prozesschritt erbt, in einem erweiterten Dialog (Abbildung 8.4(a)). Im selben Dialogfenster können auch die Daten für die Regeln zur Vor- und Nachprozessierung eingegeben werden (Abbildung 8.4(b)).

Die restriktive Vererbung für die Prozessregeln wird auf der Serverseite umgesetzt. Bei jedem Start des Konsistenzchecks (siehe Abschnitt 8.3.1) werden alle Regeln gesammelt, die der Prozessschritt besitzt und erbt. Danach werden alle Regeln geprüft. So wird sicher gestellt, dass die „schärfste“ Regel garantiert zur Anwendung kommt.

Auch die nicht restriktive Vererbung für die Parameter wird auf der Serverseite realisiert. Beim Neuanlegen eines Prozessschrittes werden alle Parameter der Prozesschritt-Typen, von denen der Prozessschritt erben soll, gesammelt. Diese Parameterdaten werden aufbereitet und dem Nutzer als Vorschlag unterbreitet. Die Aufbereitung liefert dabei das kleinste Intervall bzw. alle mögli-

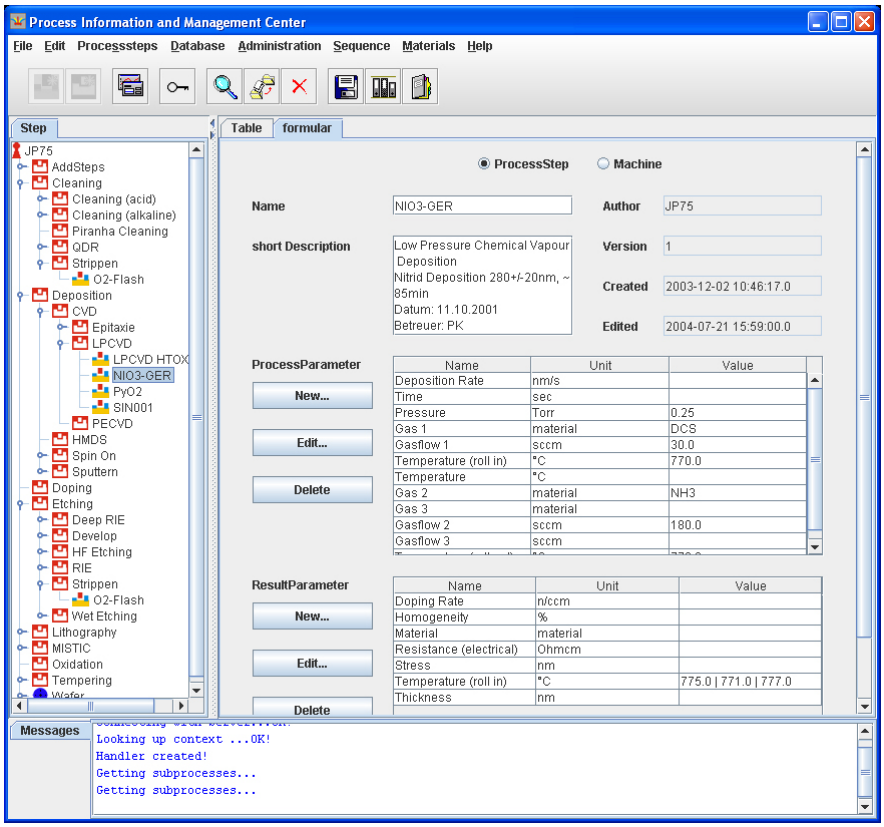
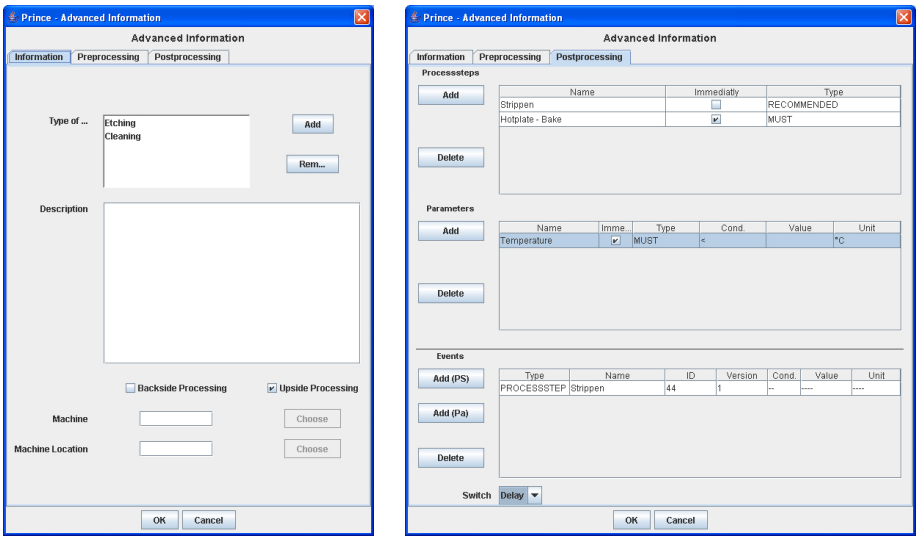


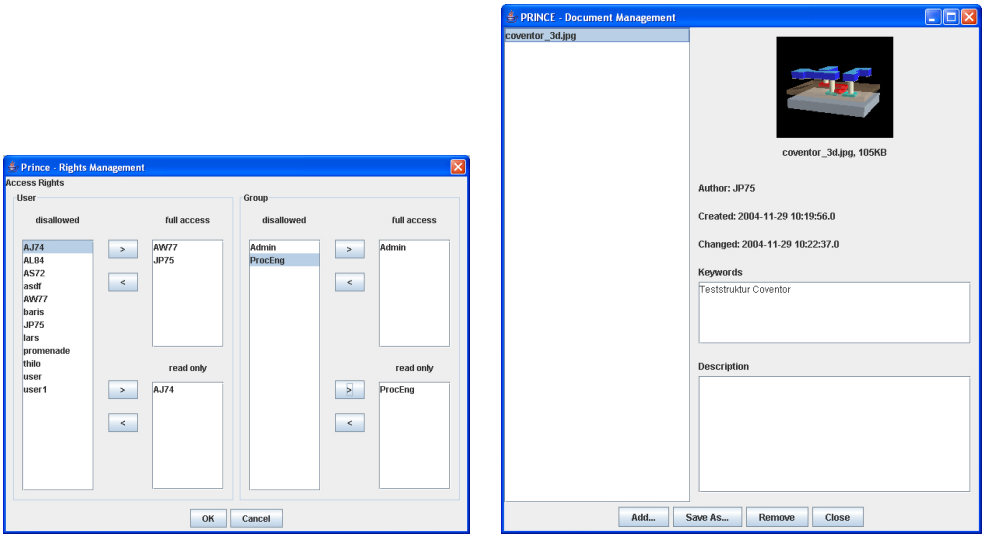
Abbildung 8.3: Prozessschritt Editor



(a) Eingabe von erweiterten Information

(b) Eingabe von Regeln

Abbildung 8.4: Erweiterter Dialog



(a) Dialogfenster für Nutzerrechte (b) Dokumentenverwaltung

Abbildung 8.5: Nutzerrechte und Dokumente für Prozessschritte

chen Werte für jeden verwendeten Parameter. Werden unterschiedliche Einheiten verwendet, so werden alle Werte in die primäre Einheit des Parameters umgerechnet.

Über das Dialogfeld in Abbildung 8.5(a) ist die Verwaltung der Rechte für einen Prozessschritt möglich. Nach der Erstellung haben auf jeden Prozessschritt nur der Nutzer selbst und Systemadministratoren Zugriff. Erst durch die Gewährung von entsprechenden Rechten wird der Prozessschritt für andere Nutzer benutzbar. Durch die Verwendung von Gruppen (Rollen) ist es möglich, mehreren Nutzern gleichzeitig den Zugriff auf die Daten zu gewähren.

Um die Dokumentation zu gewährleisten, können für jeden Prozessschritt beliebige Dokumente gespeichert werden. Abbildung 8.5(b) zeigt das entsprechende Dialogfenster. Existiert für eine Datei ein Viewer Modul (z. B. für JPEG), so wird eine Vorschau angezeigt.

8.2.1 Nutzerverwaltung

Wie bereits mehrfach erwähnt, verwaltet das PRINCE System Rechte für Nutzer und Nutzergruppen. Diese Daten können mit der Nutzerverwaltung bear-

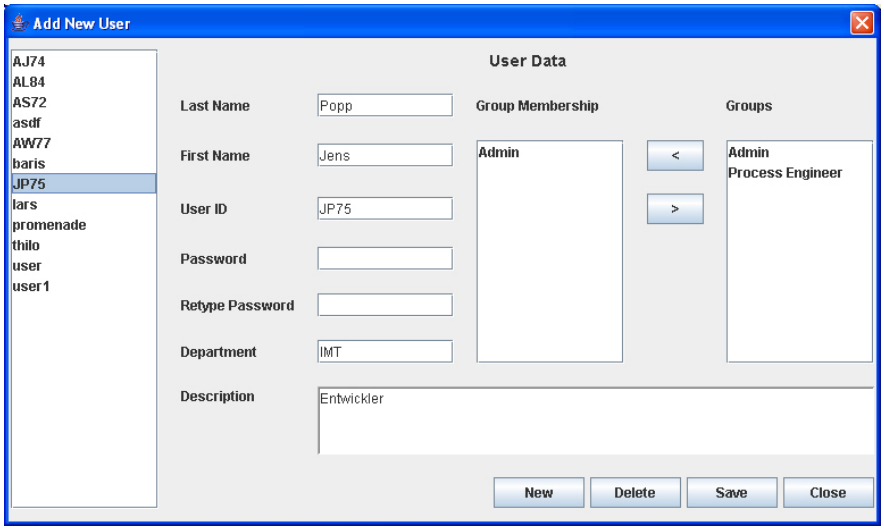


Abbildung 8.6: Nutzerverwaltung

beitet werden. Auf die Nutzerverwaltung haben nur Mitglieder der Gruppe *Admin* Zugriff. In Abbildung 8.6 wird das Dialogfenster zur Bearbeitung der Nutzerdaten angezeigt. Hier können Nutzer angelegt, geändert und gelöscht werden. Dabei können jedem Nutzer mehrere Gruppen zugewiesen werden. Abbildung 8.7 zeigt das entsprechende Fenster für die Bearbeitung von Gruppen.

8.2.2 Parameterverwaltung

Alle Parameter, die im System verwendet werden, müssen vorher über die Parameterverwaltung bekannt gemacht werden. Parameter dürfen dabei nur von Mitgliedern der Gruppe Admin angelegt oder bearbeitet werden.

In der momentanen Version der Parameterverwaltung werden alle Parameter in einer Liste dargestellt. In künftigen Versionen wird eine Baumstruktur verwendet, in der die Parameter sortiert und damit schneller aufzufinden sind. Wenn ein Parameter ausgewählt wird, so erscheinen alle Daten und Einheiten des Parameters. Über ein zusätzliches Dialogfenster können hier auch Formeln zur Umrechnung zwischen Einheiten angegeben werden (Abbildung 8.8).

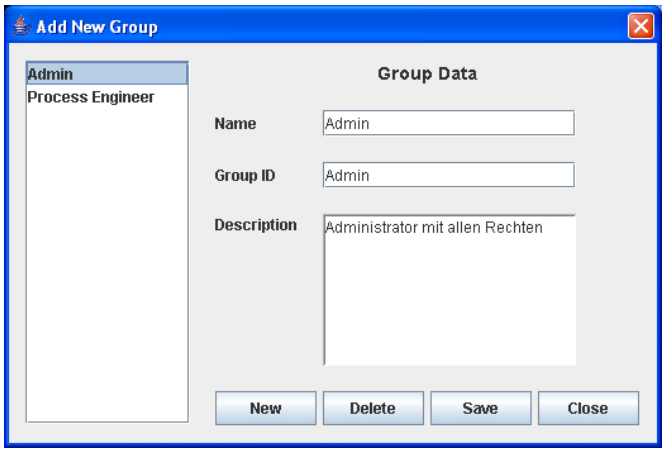
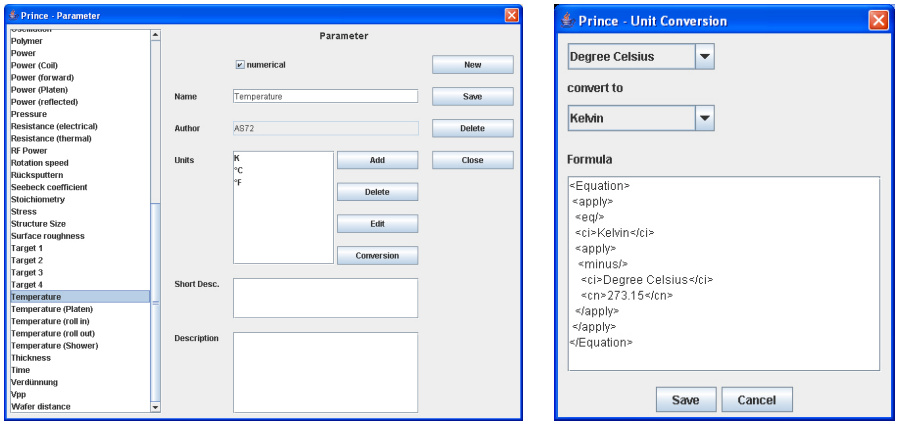


Abbildung 8.7: Verwaltung von Nutzergruppen



(a) Parameterverwaltung (b) Eingabe von Formeln

Abbildung 8.8: Verwaltung von Parametern und Einheiten

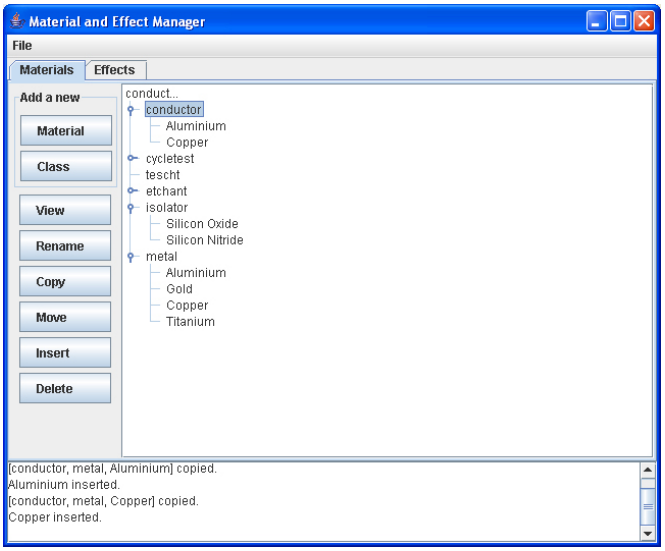


Abbildung 8.9: Materialverwaltung

8.2.3 Materialverwaltung

Materialien werden sowohl als Parameter für Prozessschritte, Schichten und Komponenten als auch als Teilnehmer an Effekten benötigt. Dabei ist es wichtig, eine einheitliche Bezugsbasis zu haben. Die Materialverwaltung bietet die Möglichkeit, Materialien zu verwalten. In Abbildung 8.9 wird die Oberfläche der Materialverwaltung gezeigt. Die Materialien sind in einer Baumstruktur angeordnet. Die Mehrfachvererbung wird wiederum für den Nutzer transparent umgesetzt. Entsprechende Materialien erscheinen in mehreren Ästen der Baumstruktur (z. B. Aluminium ist sowohl Leichtmetall als auch elektrischer Leiter).

Wird ein Material zur Bearbeitung ausgewählt, so erscheint ein Dialogfenster (Abbildung 8.10), in dem alle Daten zum Material bearbeitet werden können. Hier können auch Dokumente an die Materialien gebunden werden, die zum Beispiel aktuelle Untersuchungsergebnisse enthalten.

8.2.4 Effektverwaltung

Effekte stellen Wechselwirkungen zwischen Materialien bzw. zwischen Materialien und Prozessschritten - wie zum Beispiel die Haftung oder das Ätzen -

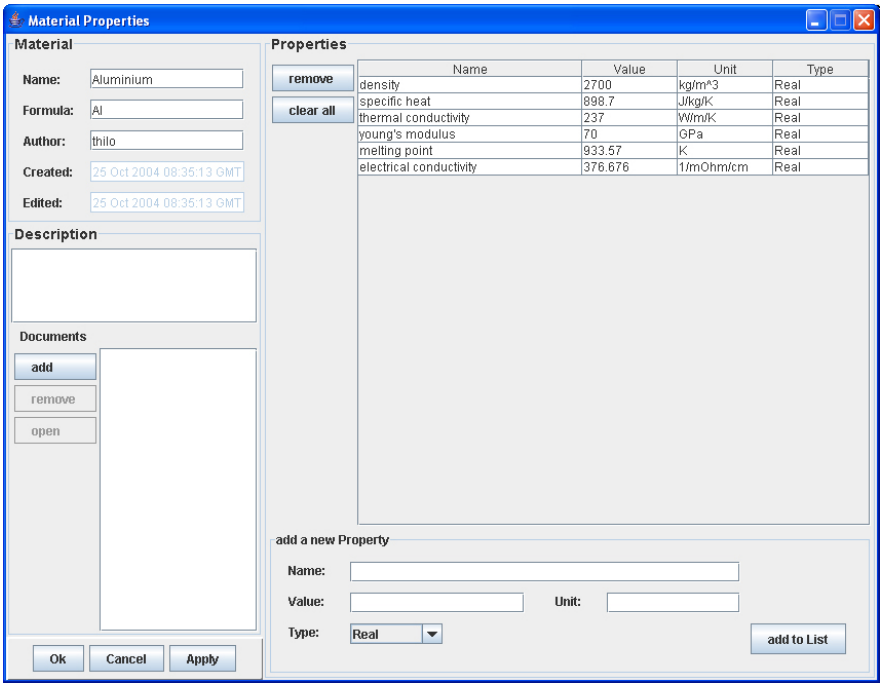


Abbildung 8.10: Eingabe von Materialdaten

dar. Die Daten die mit den Effekten verwaltet werden, können zum einen für die Konsistenzprüfung und zum anderen zur Verbesserung der Ergebnisse der Simulation verwendet werden.

Die Effekte werden in einer Baumstruktur verwaltet. Abbildung 8.11 zeigt das Fenster zur Verwaltung der Effekte. Wird ein Effekt ausgewählt, so öffnet sich ein neues Fenster mit den Daten des Effektes. Hier können Dokumente, Parameter und Materialien des Effektes verwaltet werden (Abbildung 8.12).

Über das in Abbildung 8.13 dargestellte Fenster lassen sich Materialien einem Effekt zuordnen. Dabei kann jedem Material eine Rolle zugewiesen werden (z. B. Ätzmittel, geätztes Material).

8.3 Prozessfluss Editor

Zur Verwaltung von Prozessmodulen und Prozesssequenzen wurde der Prozessfluss Editor entwickelt. Er kann aus dem Prozessschritt Editor gestartet werden (Abbildung 8.14). Mit diesem Editor kann die Prüfung der Konsistenz

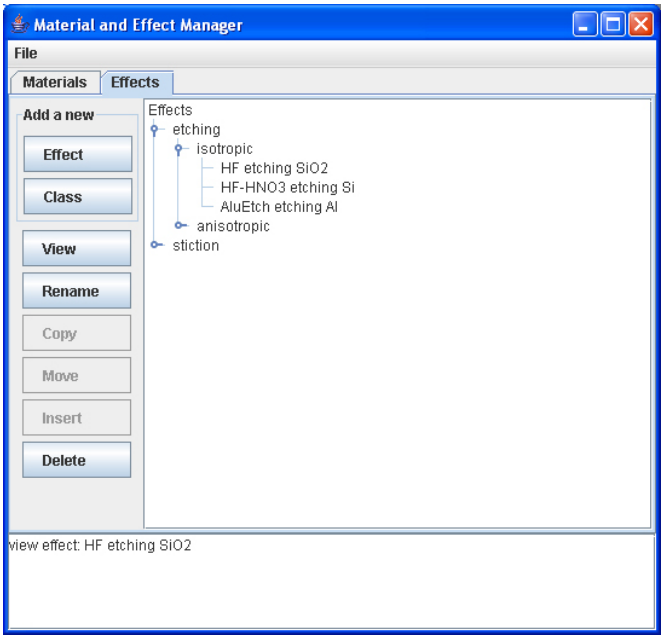


Abbildung 8.11: Effektmanager

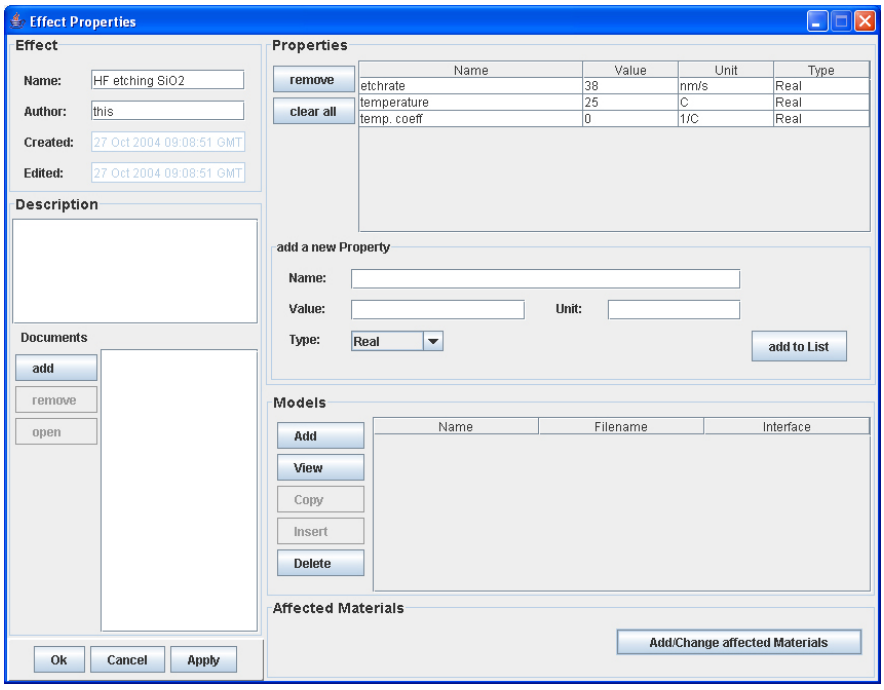


Abbildung 8.12: Eingabe von Effektdaten

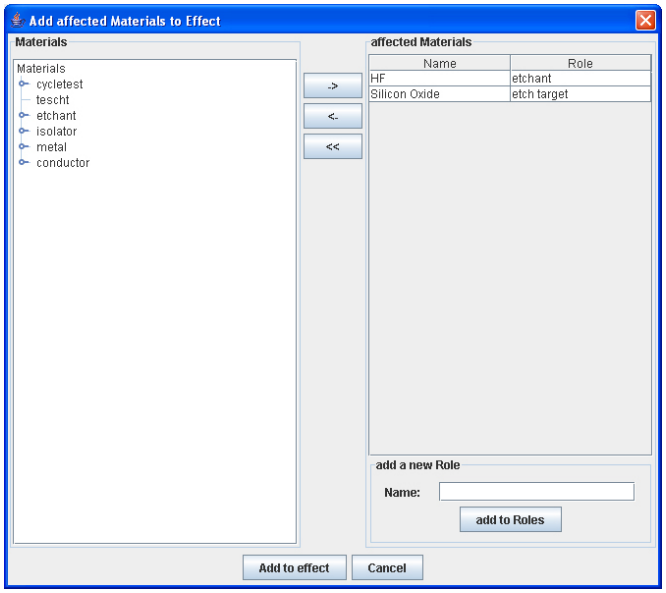


Abbildung 8.13: Zuweisung von Materialien

und die Simulation angestoßen werden. Außerdem kann eine Chargenkarte erzeugt werden.

Neue Prozessfolgen werden zusammengestellt, indem passende Prozessschritte und Prozessschrittmodule eingefügt werden. Zu diesem Zweck wurde eine Suchfunktion implementiert. Abbildung 8.15 zeigt das Dialogfenster zur Eingabe der Suchparameter. Neben den Namen und Autor können auch der Typ oder Parametergrenzen zur Einengung des Suchraumes angegeben werden.

Wird beim Öffnen einer vorhandenen Prozessfolge ein Template gewählt, so kann der Nutzer entscheiden, ob das Template bearbeitet oder eine neue Prozessfolge mit Hilfe des Templates erstellt werden soll. Ist letzteres der Fall, so öffnet sich der in Abbildung 8.16 gezeigte Dialog. Hier werden für jeden im Template verwendeten Typ alle Möglichkeiten zur Ersetzung durch reale Prozessschritte angezeigt. Auf diese Weise lässt sich sehr einfach Schritt für Schritt eine neue Prozessfolge erstellen. Die erstellte Prozessfolge kann im Nachhinein natürlich noch verändert werden.

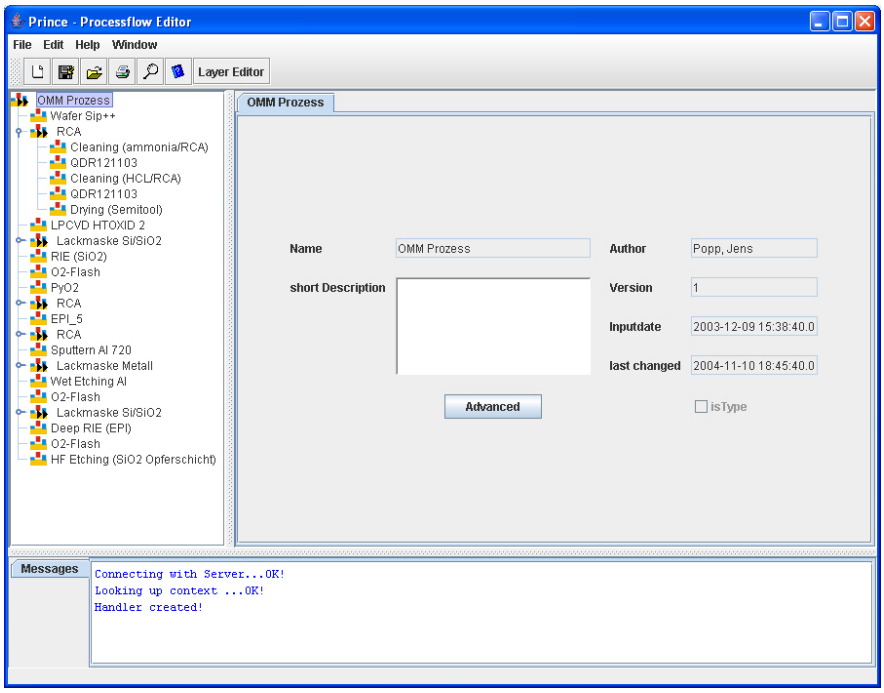


Abbildung 8.14: Prozessfluss Editor mit geöffneter Prozessfolge

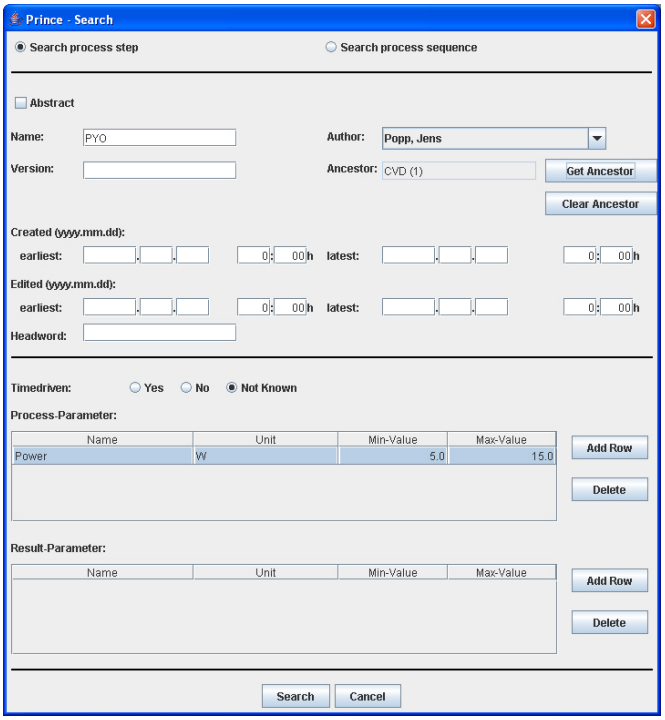


Abbildung 8.15: Dialogfenster zur Eingabe von Suchparametern

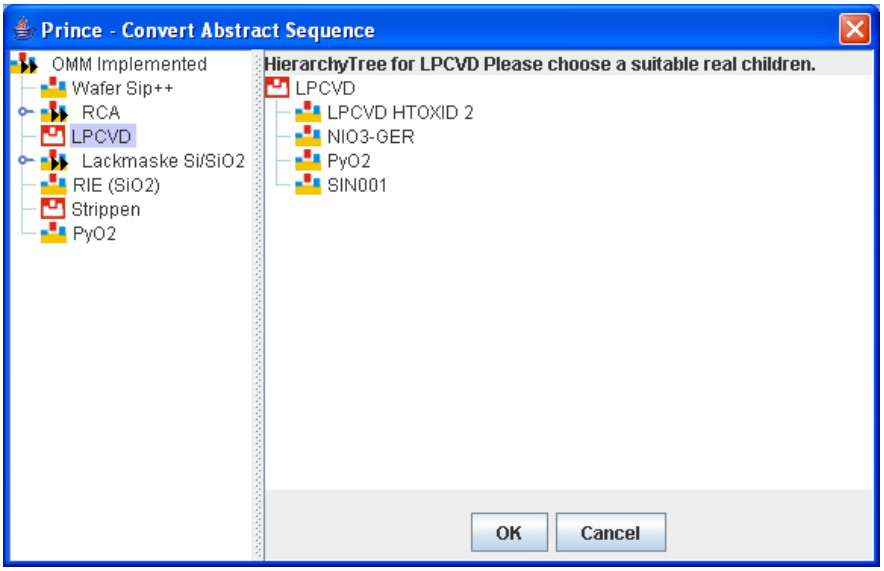


Abbildung 8.16: Ersetzen von Prozessschritttypen

8.3.1 Konsistenzcheck

Wenn eine Prozessfolge fertig zusammengestellt wurde, kann der Konsistenzcheck die Prozessfolge prüfen. Da die Prüfung selbst intensiv auf die Daten zurückgreifen muss, ist sie auf dem Server angesiedelt. Der Konsistenzcheck läuft nach folgendem Schema ab:

- Hole für jeden Prozessschritt alle Regeln zur Vor- und Nachprozessierung (auch geerbte Regeln) aus der Datenbank
- Erzeuge eine leere Liste für zu sammelnde Regeln
- Für jeden Prozessschritt X der Prozessfolge bei einem Durchlauf von vorne nach hinten:
  - Vergleiche die Parameter von X mit den gesammelten Regeln zur Nachprozessierung:
    - Gib einen Fehler aus, wenn die Regel verletzt wurde und die Regel aktiv ist
    - Lösche die Regel, wenn die Regel den Parameterwert, der mit X erreicht wurde, verlangt hat und die Regel aktiv ist

        * Deaktiviere die Regel, wenn der erreichte Parameterwert von X als ausschaltendes Event in der Regel angegeben wurde
        * Aktiviere die Regel, wenn der erreichte Parameterwert von X als einschaltendes Event in der Regel angegeben wurde
      - Prüfe, ob X oder ein Prozessschritt von dem X erbt in einer Regel zur Nachprozessierung erscheint:
        * Gib einen Fehler aus, wenn die Regel besagt, dass X oder ein Vorfahre nicht erscheinen darf und die Regel aktiv ist
        * Lösche die Regel aus dem Speicher, wenn die Regel besagt, dass X oder ein Vorfahre erscheinen muss und die Regel aktiv ist
        * Deaktiviere die Regel, wenn X oder ein Vorfahre von X als ausschaltendes Event angegeben wurde
        * Aktiviere die Regel, wenn X oder ein Vorfahre von X als einschaltendes Event angegeben wurde
      - Prüfe alle gesammelten Regeln zur Nachprozessierung, ob die zeitlichen Restriktionen (z. B. immediately) eingehalten werden, gib wenn nötig einen Fehler aus
      - Füge alle Regeln zur Nachprozessierung von X und den Vorfahren von X zu den bisher gesammelten Regeln hinzu

  - Gib für alle gesammelten Regeln zur Nachprozessierung, die noch aktiv sind und einen Prozessschritt oder die Einhaltung eines Parameterwertes verlangen einen Fehler aus

  - Wiederhole alle Schritte für Regeln zur Vorprozessierung, dabei wird die Prozessfolge von hinten nach vorne durchlaufen

Hierbei werden im ersten Schritt alle Regeln zur Vor- und Nachprozessierung gesammelt. Dabei werden nicht nur die Regeln an den Prozessschritten selbst berücksichtigt, sondern auch alle Regel an den Typen, von denen der Prozessschritt erbt. Alle für einen Prozessschritt gesammelten Regeln werden überprüft. Auf diese Weise wird immer garantiert die strikteste Regel geprüft.

Enthält eine Regel einen Typ als Vor- oder Nachprozessierung, so ist diese Regel erfüllt, wenn ein Objekt, das von diesem Typ erbt, in der Prozessfolge enthalten bzw. nicht enthalten ist. Die Vererbung ermöglicht es so, Regeln für alle

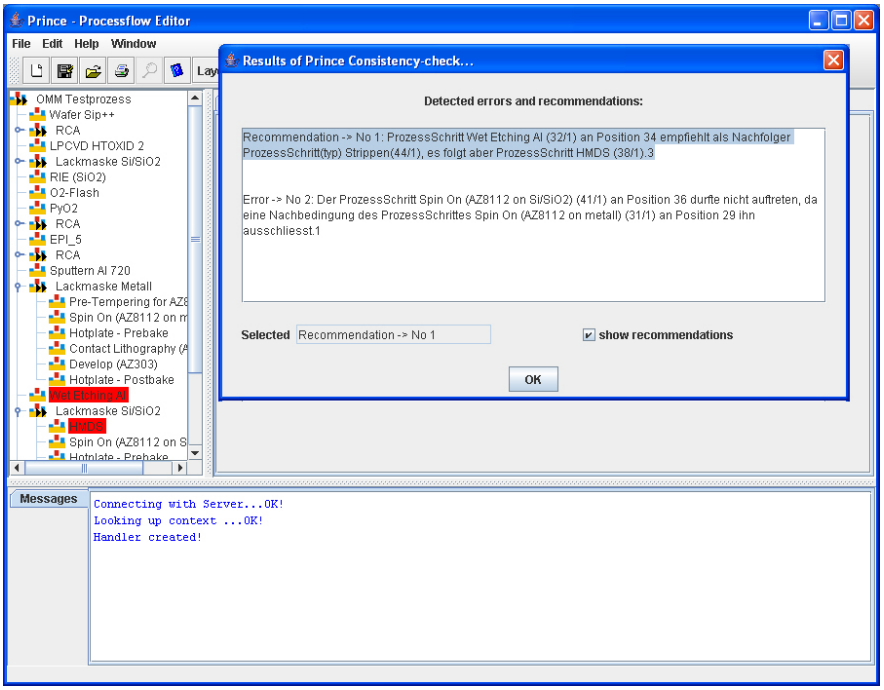


Abbildung 8.17: Ergebnis des Konsistenzchecks

Rezepte einer Klasse festzulegen. In [Stu04] wird der Algorithmus der Konsistenzprüfung genauer vorgestellt.

Die Ausgabe des Konsistenzchecks wird in Abbildung 8.17 dargestellt. Wird ein Fehler angewählt, so werden die Prozessschritte, die den Fehler verursachen im Editor farbig hervorgehoben. Auf diese Weise lässt sich schnell erkennen, wo Probleme auftreten.

8.3.2 Simulator

Im Rahmen des PRINCE Systems wurde auch ein Simulator für Prozessfolgen umgesetzt. Mit ihm ist es möglich mit Hilfe von einfachen Strichmasken einen Prozessfluss zu simulieren. Das Ergebnis einer solchen Simulation ist in Abbildung 8.18 zu sehen.

Die Ergebnisse dieser Simulation basieren auf reinen geometrischen Berechnungen und stellen keine Simulation der physikalischen Vorgänge dar. Da aber in der Realität das Ergebnis stark von physikalischen Effekten (z. B. dem Abtransport von geätztem Material) abhängt, wurde ein erweiterter Simulator

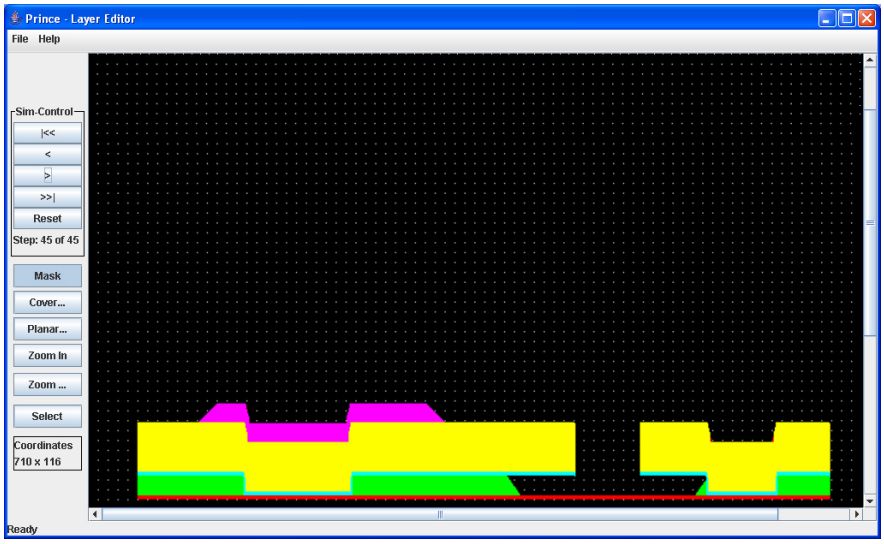
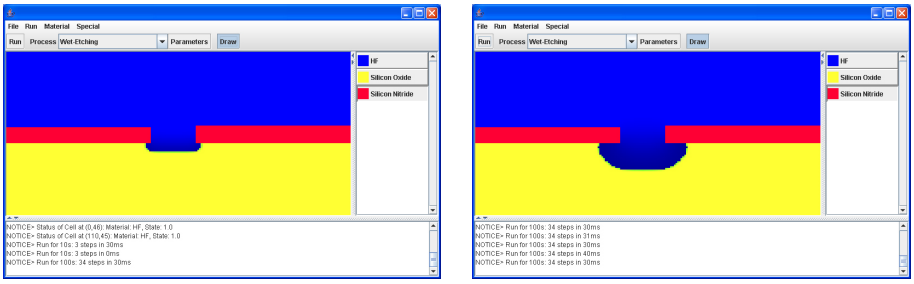


Abbildung 8.18: Ergebnis einer Simulation



(a) (b)

Abbildung 8.19: Ätzsimulator mit zellulärem Automaten

entwickelt [Sch04a, Wag05]. Hier werden die mit den Effektdaten gespeicherten Modelle verwendet, um einen zellulären Automaten zu initialisieren. Mit Hilfe dieser Modelle können auch Effekte wie die Diffusion von geätztem Material in engen Kanälen simuliert werden. Abbildung 8.19 zeigt das Ergebnis einer Ätzsimulation mit diesem Simulator. Das Ätzmedium (hier blau) wird dabei umso dunkler, je mehr abgetragenes Material in ihm enthalten ist. Dies bewirkt, das der Ätzvorgang verlangsamt wird.

8.4 Nutzungsszenarien

Während der Entwicklung des PRINCE-Systems wurde eng mit den späteren Nutzern zusammengearbeitet. In mehreren Einzel- und Gruppengesprächen wurden Anforderungen und Nutzungsszenarien erarbeitet. Dabei wurde stets darauf geachtet, dass den Untersuchungen reale Daten zu Grunde lagen. Im Folgenden sollen die Nutzungsszenarien und ihre Umsetzung in der Software kurz vorgestellt werden.

Administrator

Der Administrator ist verantwortlich für die Verwaltung des Systems. Nur der Administrator kann Parameter, Einheiten und Nutzer anlegen, verändern oder löschen. Er hat außerdem Zugriff auf alle im System gespeicherten Daten. Deshalb hat der Administrator eine sehr verantwortungsvolle Tätigkeit. In späteren Versionen ist angedacht, die Aufgaben auf verschiedene Administratoren zu verteilen. So können für jede Arbeitsgruppe oder für jeden Aufgabenbereich eigene Administratoren mit beschränkten Rechten ernannt werden.

Die Einführung der Datenhaltung hat wesentlichen Einfluss auf die Arbeit des Administrators. Erst durch die Datenhaltung wird die zentrale Verwaltung von Rechten und Parametern ermöglicht. Die Einführung der Vererbung erleichtert den Umgang mit den Daten und die Verteilung von Rechten. Dem Administrator werden mit den bereits in Abschnitt 8.2 vorgestellten Dialogen zur Verwaltung von Nutzern und Parametern die entsprechenden Werkzeuge zur Verfügung gestellt.

Prozessschritt-Entwickler

Der Prozessschritt-Entwickler ist verantwortlich für die Entwicklung von speziellen Prozessschritten. Hierbei geht es meist um die Optimierung von Schichtparametern. Dabei kann der Prozessschritt-Entwickler erheblich von der Datenhaltung profitieren.

Die Suchfunktion ermöglicht es, nach bereits durchgeführten Experimenten zu suchen, die eventuell bereits die Zielparameter beinhalten. Ist dies der Fall, so

kann viel Zeit bei der Entwicklung gespart werden. Aber auch bei der Neuentwicklung hilft die organisierte Datenhaltung. Rezepte, die im PRINCE-System verwaltet werden, können von jedem Rechner im Intranet aus eingesehen werden. So kann der Entwickler im Reinraum die Daten für die Maschine entnehmen, im Labor Messungen durchführen und entstehende Dokumente (z. B. Mikroskopbilder) einfügen und im Büro die Ergebnisse auswerten und Analysen in das System eintragen.

Durch die Vererbung können Schablonen für Versuchsserien angelegt werden. Der Entwickler kann dann von diesen Schablonen neue Rezepte ableiten, in denen nur noch die geänderten Parameter angepasst werden müssen. Darüber hinaus kann der Entwickler neu erkannte Regeln an den Schablonen festmachen. Alle Rezepte die von dieser Schablone erben, erhalten dann automatisch die entsprechenden Regeln.

Dem Entwickler steht für diese Aufgabe der Prozessschritteditor (siehe Abschnitt 8.2) zur Verfügung. Hier gibt es die Möglichkeit Prozess- und Ergebnisparameter einzugeben, Dokumente und Regeln zu verwalten und Vererbungshierarchien aufzubauen.

Prozessfluss-Entwickler

Der Prozessfluss-Entwickler erstellt komplette Prozesssequenzen für neue Produkte oder verbessert bereits vorhandene. Dabei nutzt er zumeist die vom Prozessschritt-Entwickler erarbeiteten Prozessschritte beziehungsweise arbeitet eng mit ihm zusammen, um neue Prozessschritte zu erstellen[1]. Dabei verwendet er die meisten anderen Prozessschritte nur, ohne eine tiefere Kenntnis über diese zu besitzen.

Der Prozessfluss-Entwickler kann sehr stark von der Datenhaltung profitieren. Durch die Verwendung der PRINCE-Software kann er auf alle Daten der Prozessschritt Entwickler zugreifen, für die er entsprechende Rechte besitzt. Mit der Suchfunktion kann er so zum Beispiel auch Prozessschritte finden, mit denen er persönlich noch keine Erfahrung gemacht hat.

[1] Häufig ist in letzteren Fall der Prozessschritt- und der Prozessfluss-Entwickler ein und dieselbe Person.

Für die Erstellung neuer Prozessfolgen ist die Nutzung der Vererbung zur Bereitstellung von Schablonen sehr nützlich. Auf diese Weise können Schablonen für Versuchsserien entwickelt werden, bei denen in der abgeleiteten realen Prozessfolge nur die kritischen Prozessschritte ersetzt werden.

Eine der größten Vorteile für den Prozessfluss-Entwickler ist jedoch der Konsistenzcheck. Da der Prozessschritt-Entwickler bereits Regeln für seine Prozessschritte entwickelt hat, kann der Prozessfluss-Entwickler diese nutzen, um seine neu entwickelte Folge zu prüfen. So können teure Fehler wie ein vergessener Reinigungs- oder Annealingschritt vermieden werden.

Mit der Simulation kann der Prozessfluss-Entwickler schließlich im vorab überprüfen, wie das Ergebnis der Prozessfolge aussehen könnte (je nach Güte der verwendeten Simulationsmodelle). Auch hier können Fehler bereits früh erkannt und teure Fehlläufe in der Fertigung vermieden werden. Mit dem Prozessfluss Editor (siehe Abschnitt 8.3) werden dem Entwickler all diese Werkzeuge in die Hand gegeben.

Device-Entwickler

Der Device-Entwickler führt für vorhandenen Prozessfolgen die geometrischen Designs (z. B. Maskenlayout) durch. Mit dem Device-Entwickler wird ein Nutzer eingeführt, der normalerweise mit dem fertigungsnahen Entwurf nur wenig zu tun hat. Trotzdem kann gerade er von einer guten Datenhaltung profitieren.

Der Device-Entwickler arbeitet mit Simulatoren, die das Verhalten des künftigen Produktes untersuchen sollen. Dabei ist er auf exakte Daten über verwendete Materialien, Schichtdicken, Stress und andere Größen angewiesen. Bisher wurden diese Werte meist empirisch in Experimenten für jedes Device neu gewonnen. Mit Verwendung der Datenhaltung können einmal ermittelte Werte gespeichert und immer wieder verwendet werden. Außerdem kann durch die Verwendung von Prozess-Simulatoren überprüft werden, wie sich zum Beispiel die Änderung eines Parameters auf eine Schichtfolge auswirkt. Die Ergebnisse können dann extrahiert und in anderen Simulatoren zur Bestimmung des Verhaltens verwendet werden.

Ein anderer Vorteil der Datenhaltung in Verbindung mit Prozess-Simulatoren ist die Extraktion von Design Regeln für Prozessfolgen. So ist es - in Grenzen - möglich, einfache Regeln für die Erstellung von Masken aus der Prozessfolge zu erzeugen.

Die Anwendung des PRINCE-Systems zur Unterstützung des Device-Entwicklers ist noch am wenigsten fortgeschritten. Bisher wurden Prozess-Simulatoren genutzt, um Strukturen (wie z. B. Schrägen nach dem Ätzen) zu erkennen. Die Daten in Zusammenhang mit den Daten aus der Materialdatenbank können zur Initialisierung einer Verhaltenssimulation dienen. Eine verbesserte Integration der Prozess-Simulatoren und die Extraktion von relevanten Werten aus den Simulationsergebnissen ist ein Ziel des aktuellen EU-Projektes Promenade.

9 Resümee und Ausblick

In dieser Arbeit wurde ein Konzept zur Datenhaltung für die fertigungsnahe Mikrosystemtechnik vorgestellt. Dazu wurden bisherige Ansätze zur Lösung dieses Problems betrachtet und bewertet. Gute Ideen, wie die Verwendung eines Datenbanksystems und die Einführung der Vererbung in die Datenhaltung, wurden übernommen, angepasst und erweitert.

Durch die Untersuchung aktueller Entwurfsmethoden wurden Softwaremodule identifiziert, die eine Datenhaltung benötigen. Dabei wurde vor allem festgestellt, dass alle Daten stark miteinander verwoben sind. Alle Daten der Materialien, Effekte und aus der Fertigung müssen also in enger Beziehung zueinander gesehen werden. Die Darstellung von unterschiedlichen Klassifizierungen der Daten wurde durch die Einführung der Mehrfachvererbung ermöglicht.

Die Untersuchung aktueller Technologien zur Speicherung von Daten und dazu passender Softwarearchitekturen führte zur Auswahl einer J2EE-basisierten Lösung mit einer relationalen Datenbank als Datenspeicher. Die gesammelten Daten wurden für diese Architektur formalisiert und im Softwaresystem PRINCE realisiert.

Mit PRINCE wurde ein Softwaresystem eingeführt und in der Praxis getestet, das das Management der Daten ermöglicht. Die Erstellung, Verwaltung und die Prüfung von Prozessfolgen wird unterstützt. Durch die Möglichkeit, Daten zentral zu verwalten, haben alle Entwickler Zugriff auf den aktuellen Datenbestand. So ist bereits jetzt durch die Verwendung der Software die Einsparung von Zeit und somit auch Kosten möglich.

Teilergebnisse der hier und in [Wag05] vorgestellten Arbeit wurden auf nationalen [WPHB04] und internationalen [WPHB02a, WPHB02b, WPH03a, HW03, WPH03b, WPH04, SWPH04, SWPH05] Konferenzen vorgestellt. Die Relevanz

dieser Arbeit zeigt sich auch in der Tatsache, dass ein Folgeprojekt von der EU mit mehreren Millionen Euro gefördert wird. Dabei fließen die Erfahrungen, die bei der praktischen Anwendung des PRINCE Systems gemacht wurden, in das EU-Projekt Promenade ein.

Zielsetzungen in Promenade sind die Weiterentwicklung der Software und die Anbindung anderer Software an die Datenhaltung. Durch die Verwendung der gesammelten Daten zur Simulation von Prozessschritten können die Ergebnisse der Simulation verbessert werden. Diese Ergebnisse können wiederum in die Datenhaltung fließen und für Simulationen des Verhaltens genutzt werden.

Neben der Anbindung von Simulatoren ist ein weiteres Ziel die Optimierung von Prozessfolgen. Durch Austauschen von Prozessmodulen und geschickte Suche in der Datenbank können Prozessfolgen nach bestimmten Kriterien optimiert werden. In Zusammenhang mit einer Generierung von Prozessfolgen, wie sie rudimentär mit Mistic vorgestellt wurde, kann so eine umfassende Unterstützung des fertigungsnahen Entwurfes von Mikrosystemen angeboten werden. Ohne die Verwendung einer umfassenden Datenhaltung, wie sie in dieser Arbeit vorgestellt wurde, wäre dies nicht möglich.

Abbildungsverzeichnis

10 Literaturverzeichnis

[AT00] ANDERL, R. ; TRIPPNER, D.: *STEP STandard for the Exchange of Product Model Data*. Stuttgart, Leipzig : B. G. Teubner, 2000. – ISBN 3–519–06377–8

[Aut04] AUTODESK GMBH. *Autodesk Home Page*. http://www.autodesk.de/. November 2004

[Bal03] BALES, M.: FACILITATING EDA FLOW INTEROPERABILITY WITH THE OPENACCESS DESIGN DATABASE. In: *Proceedings of the Electronic Design Processes Conference 2003*, 2003

[Brü96] BRÜCK, Rainer. *Der fertigungsnahe Entwurf von Mikrosystemen*. Habilitation, Universität Dortmund. 1996

[Bro04] BROOKS AUTOMATION, INC. *PROMIS - Configurable Manufacturing Execuition System*. http://www.brooks.com/pages/215_promis.cfm. November 2004

[BRS01a] BRÜCK, Rainer (Hrsg.) ; RIZVI, Nadeem (Hrsg.) ; SCHMIDT, Andreas (Hrsg.): *Angewandte Mikrotechnik*. 1. Aufl. München Wien : Hanser, Jan. 2001

[BRS01b] BRÜCK, Rainer ; RIZVI, Nadeem ; SCHMIDT, Andreas: *Angewandte Mikrotechnik: LIGA - Laser - Feinwerktechnik*. Hanser Verlag, 2001

[Che76] CHEN, P.: The entity-relationship model: Toward a unified view of data. In: *ACM Transactions on Database Systems, 1*, 1976

[Cod70] CODD, E.F.: A relational model for large shared databanks. In: *Communications of the ACM, 13*, 1970

[Cod79] CODD, E.F.: Extending the relational model to capture more meaning. In: *ACM Transactions on Database Systems, 4*, 1979

[Con99] CONRADI, P.: *Reuse in Electronic Design*. Chinchester, New York, Weinheim, Brisbane, Singapore, Toronto : John Wiley & Sons, 1999. – ISBN 0–471–9750–6

[Cov04] COVENTOR, INC. *Coventor Home Page*. http://www.coventor.com/. November 2004

[Das03] DASGUPTA, Sumit: OpenAccess: Goals and Status. In: *Proceedings of the Design Automation and Test in Europe Conference 2003*, 2003

[Doc] DOCUMENTUM. *Enterprise Content Management*. http://www.documentum.com

[Ele04] ELECTRICAL ENGINEERING AND COMPUTER SCIENCES, UNIVERSITY OF CALIFORNIA, BERKELEY. *SIMPL-er*. http://www.ocf.berkeley.edu/~hhile/SIMPLer/. April 2004

[EN00] ELMASRI, Ramez ; NAVATHE, Shamkant B.: *Fundamentals of database systems*. 3rd edition. Addison Wesley, 2000. – ISBN 0–8053–1755–4

[Eur04] EUROPEAN COMMISSION. *PROMENADE - Process management and design system for microsystem technologies*. Specific targeted research or innovation project, Contract 507965, IST-2002-2.3.1.2. 2004

[FR97] FATIKOW, S. ; REMBOLD, U.: *Microsystem Technology and Microrobotics*. Springer-Verlag Berlin Heidelberg, 1997

[GK83] GAJSKI, D. D. ; KUHN, R. H.: New VLSI Tools - Guest Editor's Introduction. In: *IEEE Computer* Bd. 16. 1983

[Hah98a] HAHN, Kai: *Methoden und Werkzeuge zur fertigungsnahen Entwurfsverifikation in der Mikrotechnik*, Universität Siegen, Diss., 1998

[Hah98b] HAHN, Kai: Die Prozessbeschreibungssprache LIDO-PDL / Universität Dortmund. 1998 (695). – Forschungsbericht

[HB01] HANSEN, Ulli ; BÜTTGENBACH, Stephanus: A Data Model for the Representation of Fabrication Dependencies concernign Micromechanical Devices. In: *Modeling and Simulation of Microsystems*, 2001, S. 586–589

[HGBF03] HANSEN, U. ; GERMER, C. ; BÜTTGENBACH, S. ; FRANKE, H.J.: Mixed System and Component Level T-CAD for Micro Fabrication. In: *DTIP of MEMS and MOEMS*, 2003

[HTB+03] HANSEN, U. ; TRILTSCH, U. ; BÜTTGENBACH, S. ; GERMER, C. ; FRANKE, H.J.: Analysis and Verification of Process Sequences. In: *Modeling and Simulation of Microsystems*, 2003

[HW03] HAHN, K. ; WAGENER, A.: *Considerations for MEMS physical design stages*. Invited Tutorial, Sophia Antipolis Microelectronics, Sophia Antipolis. 2003. – SAME 2003

[Hyp04] HYPERWAVE AG. *HYPERWAVE - The power of Wisdom - Content Management*. http://www.hyperwave.com/. April 2004

[Ins04a] INSTITUT FÜR MIKROTECHNIK MAINZ GMBH. *Institut für Mikrotechnik Mainz*. http://www.imm-mainz.de/. November 2004

[Ins04b] INSTITUT FÜR MIKROSYSTEMTECHNIK: *PRINCE - Benutzerhandbuch*. Universität Siegen, Dezember 2004

[Kle02] KLEINERT, Andreas: *Entwicklung einer Prozessbeschreibungssprache für die Mikrosystemtechnik auf Basis von XML*, Universität Siegen, Diplomarbeit, 2002

[Lee85] LEE, K.: *SIMPL - 2 (SIMULATED PROFILES FROM THE LAYOUT) VERSION 2*, University of Berkeley, Diss., 1985

[LN04] LENTZER, A. ; NEUL, R.: Umfrage zum Werkzeugeinsatz und Werkzeugebedarf beim Entwurf von Mikrosystemen. / ZVEI Zentralverband Elektrotechnik- und Elektronikindustrie e.V. 2004. – Forschungsbericht

[Mad02] MADOU, M.: *Fundamentals of Microfabrication*. 2nd edition. CRC Press, 2002

[MC80] MEAD, Carver ; CONWAY, Lynn: *Introduction to VLSI systems*. Addison-Wesley, Reading, Massachusetts, 1980

[Mes00] MESCHEDER, U.: *Mikrosystemtechnik, Konzepte und Anwendungen*. B.G. Teubner, 2000

[MW00] MEIER, Andreas ; WÜST, Thomas: *Objektorientierte und objektrelationale Datenbanken: Ein Kompaß für die Praxis*. 2. Auflage. Heidelberg, Germany : dpunkt.verlag, 2000. – ISBN 3–932588–68–1

[Obj04] OBJECT DATA MANAGEMENT GROUP. *ODMG Home Page*. http://www.odmg.org/. November 2004

[Ora04] ORACLE CORPORATION. *Oracle Corporation*. http://www.oracle.com/. November 2004

[Pri04] PRIEBE, Andreas: *Innovative Softwareunterstützung für die Wissensvermittlung in Ingenieurwissenschaften*, Universität Siegen, Diss., 2004

[PWOB04] POPP, J. ; WAGENER, A. ; ORTLOFF, D. ; BEUNDER, M.: A Novel Approach Towards Standardization of MEMS Process Development. In: *Proceedings of the COMS 2004*, 2004. – COMS 2004

[Rol03] ROLLAND, F.D.: *Datenbanksysteme im Klartext*. Paderborn, Germany : Pearson Studium, 2003. – ISBN 3–8273–7066–3

[Sch04a] SCHMIDT, Thilo: *Konzeption und Implementierung einer Material- und Effektdatenbank für Simulation und Analyse in der Mikrosystemtechnik*, Universität Siegen, Diplomarbeit, 2004

[Sch04b] SCHUMER, Christian: *Design Flow Management mit Web Services in der Mikrotechnik*, Universität Siegen, Diss., 2004

[Sem04] SEMICONDUCTOR, MEMS AND FDP INDUSTRY. *Semiconductor, MEMS and FDP Industry Information*. http://www.semi.org/. November 2004

[Sil04a] SILICON INTEGRATION INITIATIVE, INC. *Silicon Integration Initiative*. http://www.si2.org/. November 2004

[Sil04b] SILVACO. *Silvaco Website*. www.silvaco.com. Februar 2004

[STE04] STEAG MICROPARTS GMBH. *Kleine Teile für große Ziele*. http://www.microparts.de/. November 2004

[Stu04] STUFF, Alexander: *Entwicklung und Implementierung eines Algorithmus zur Überprüfung und Sicherstellung der Konsistenz von Prozessfolgen in der Mikrosystemtechnik*, Universität Siegen, Diplomarbeit, 2004

[Sun04a] SUN MICROSYSTEMS. *Java Remote Method Invocation*. http://java.sun.com/products/jdk/rmi/. November 2004

[Sun04b] SUN MICROSYSTEMS. *Java Technology*. http://java.sun.com/. November 2004

[Sun04c] SUN MICROSYSTEMS. *JDBC Technology*. http://java.sun.com/products/jdbc/. November 2004

[Sun04d] SUN MICROSYSTEMS INC. *Java 2 Platform, Enterprise Edition (J2EE) Homepage*. http://java.sun.com/j2ee/. November 2004

[Sun04e] SUN MICROSYSTEMS INC. *Java Web Start Homepage*. http://java.sun.com/products/webstart/. November 2004

[SWPH04] SCHMIDT, T ; WAGENER, A. ; POPP, J. ; HAHN, K.: Technology Interfaces to Microsystem and Nanoelectronic Processes. In: *Smart Structures, Devices and Systems II* Bd. 5649, 2004

[SWPH05] SCHMIDT, T. ; WAGENER, A. ; POPP, J. ; HAHN, K.: MEMS fabrication process management environment. In: *Micromachining and Microfabrication Process Technology X* Bd. 5715, 2005

[TP04] TRANSTEC-PROJECT. *TRANSTEC Internet-based Multimedia Knowledge Transfer for Innovative Engineering Technologies*. http://www-ttec.rs.uni-siegen.de/. April 2004

[Tro04] TROLLTECH. *Trolltech - Creators of Qt*. http://www.trolltech.com. November 2004

[Wag01] WAGENER, Andreas: *System- und Anforderungsanalyse für ein Prozessdesign-Werkzeug in der Mikrosystemtechnik*, Universität Siegen, Diplomarbeit, 2001

[Wag05] WAGENER, Andreas: *Eine fertigungsnahe Entwurfsunterstützung für die Mikrosystemtechnik*, Universität Siegen, Diss., 2005

[WG04] WREGE, Jan ; GRISCHEK, Viola. *SFB516 Konstruktion und Fertigung aktiver Mikrosysteme*. http://www.sfb516.tu-bs.de/. April 2004

[Wor04a] WORLD WIDE WEB CONSORTIUM (W3C). *Extensible Markup Language (XML) Homepage*. http://www.w3.org/XML/. November 2004

[Wor04b] WORLD WIDE WEB CONSORTIUM (W3C). *W3C Math Home*. http://www.w3.org/Math/. November 2004

[WPH03a] WAGENER, A. ; POPP, J. ; HAHN, K.: PRINCE - Process Information and Management Center. In: *Proceedings Micro System Technologies 2003, München*, 2003. – MST 2003

[WPH03b] WAGENER, A. ; POPP, J. ; HAHN, K.: Process management and design for MEMS and microelectronics technologies. In: *Microelectronics: Design, Technology, and Packaging* Bd. 5274, 2003

[WPH04] WAGENER, A. ; POPP, J. ; HAHN, K.: Process design environment for microfabrication technologies. In: *Micromachining and Microfabrication Process Technology IX* Bd. 5342, 2004

[WPHB02a] WAGENER, A. ; POPP, J. ; HAHN, K. ; BRÜCK, R.: PDML - A XML-Based Process Description Language. In: *Proceedings of the 9th European Concurrent Engineering Conference, Modena*, 2002. – ECEC 2002

[WPHB02b] WAGENER, A. ; POPP, J. ; HAHN, K. ; BRÜCK, R.: Requirements to a physical design support tool for microsystem technology. In: *Proceedings of the 14th European Simulation Symposium, Dresden*, 2002. – ESS 2002

[WPHB04] WAGENER, A. ; POPP, J. ; HAHN, K. ; BRÜCK, R.: PRINCE - Design Tools for Micro System Fabrication Management. In: *10. GMM Workshop: Methoden und Werkzeuge für den Entwurf von Mikrosystemen*, 2004

[WT85] WALKER, R. A. ; THOMAS, D. E.: A Model for Design Representation and Synthesis. In: *Proceedings of the Design Automation Conference*, 1985

[Wu88] WU, Hsi-Cheng: *SIMPL - DIX (SIMulated Profiles from the Laout - Design Interface in X)*, University of Berkeley, Diss., 1988

[ZCM99a] ZAMAN, Mohammed H. ; CARLEN, Edwin T. ; MASTRANGELO, Carlos H.: Automatic Generation of Thin Fil Process Flows - Part I: Basic Algorithms. In: *IEEE Transactions on Semiconductor Manufacturing* Bd. 12 No. 1 IEEE, 1999

[ZCM99b] ZAMAN, Mohammed H. ; CARLEN, Edwin T. ; MASTRANGELO, Carlos H.: Automatic Generation of Thin Fil Process Flows - Part II: Recipe Generation, Flow Evaluation, and System Framework. In: *IEEE Transactions on Semiconductor Manufacturing* Bd. 12 No. 1 IEEE, 1999

[ZD02] ZHA, Xuan F. ; DU, H.: Web-based knowledge-intensive support framework for collaborative design of MEMS. In: *Journal of Micromechanics and Microengineering* Bd. 12. UK : Institute of Physics Publishing, 2002, S. 512–524

[ZD03] ZHA, Xuan F. ; DU, H.: Manufacturing process and material selection in concurrent collaborative design of MEMS devices. In: *Journal of Micromechanics and Microengineering* Bd. 13. UK : Institute of Physics Publishing, 2003, S. 509–522