Forschungszentrum Karlsruhe
Technik und Umwelt

Wissenschaftliche Berichte
FZKA 5843

MIDAS

Ein wissensbasiertes System zur Unterstützung des fertigungsgerechten Entwurfs von LIGA-Mikrostrukturen

P. Buchberger, H. Eggert, K. P. Scherer

Institut für Angewandte Informatik
Projekt Mikrosystemtechnik

Februar 1997

Forschungszentrum Karlsruhe
Technik und Umwelt

Wissenschaftliche Berichte
FZKA 5843

M I D A S

Ein wissensbasiertes System zur Unterstützung des fertigungsgerechten Entwurfs von LIGA-Mikrostrukturen

P. Buchberger*), H. Eggert, K.P. Scherer
Institut für Angewandte Informatik
Projekt Mikrosystemtechnik

*)vom Fachbereich 1 Physik/Elektrotechnik
der Universität Bremen genehmigte Dissertation

Forschungszentrum Karlsruhe GmbH, Karlsruhe
1997

Als Manuskript gedruckt

Forschungszentrum Karlsruhe GmbH
Postfach 3640, 76021 Karlsruhe

ISSN 0947-8620

MIDAS
Ein wissensbasiertes System zur Unterstützung des fertigungsgerechten Entwurfs von LIGA-Mikrostrukturen

Zusammenfassung

Das zur Mikrostrukturierung entwickelte LIGA-Verfahren befindet sich derzeit an der Schwelle zu einer industriellen Nutzung. Hierdurch unterliegt diese Technologie in verstärktem Maße auch ökonomischen Randbedingungen. Für eine optimale Nutzung der Fertigungspotentiale ist die Gestaltung der Mikrostrukturen von zentraler Bedeutung, da sie Auswirkungen auf den gesamten Fertigungs- und Montageprozeß hat. Daher ist es wichtig, technologische Einflüsse auf die Produktgestalt bereits bei der Konstruktion zu berücksichtigen.
Hierzu ist der Konstrukteur auf ein entsprechendes Hintergrundwissen über die zugrundeliegende Herstellungsmethode angewiesen. Das umfangreiche zur Konstruktion von LIGA-Strukturen benötigte Fertigungswissen hat inzwischen jedoch eine Komplexität erreicht, die nicht mehr ohne eine angemessene informationstechnische Unterstützung handhabbar ist. Ein entsprechendes Werkzeug soll dem Konstrukteur mit dem im Laufe dieser Arbeit entwickelten wissensbasierten System MIDAS (Microstructure Diagnosis and Answerbook System) an die Hand gegeben werden.
Der erste Schritt zu einer auf Fertigungswissen basierenden Konstruktionsunterstützung ist, dem Konstrukteur einen für die Entwurfsvorbereitung geeigneten interaktiven Zugang zu den von ihm benötigten Informationen zu ermöglichen und Mittel zu ihrer Erfassung bereitzustellen. Bei MIDAS wurde dies in Form eines auf objektorientierten Modellen basierenden Informationssystems realisiert.
Einen Schritt weiter geht das Konzept einer automatischen Entwurfsdiagnose, wie sie bereits in der Mikroelektronik als Standardvalidierungsschritt üblich ist. Ziel einer Entwurfsdiagnose im Sinne eines fertigungsgerechten Entwurfes ist es, die Einhaltung fertigungstechnischer Randbedingungen hinsichtlich Form und Abmessungen der herzustellenden Mikrostrukturen zu gewährleisten. MIDAS stellt hierfür ein Diagnosewerkzeug zur Verfügung, das die fertigungsgerechte Gestaltung von Entwürfen auf der Basis eines geeigneten Satzes von Gestaltungsregeln überprüft. Hierbei werden die Werte der davon betroffenen Gestaltparameter aus der Entwurfszeichnung extrahiert und mit den fertigungstechnisch sinnvollen Wertebereichen verglichen. Verletzungen dieser Grenzwerte werden vom Diagnosesystem protokolliert. Den Abschluß der Diagnose bildet die Interpretation des Fehlerprotokolles im Dialog mit dem Konstrukteur.
Das im Laufe dieser Arbeit entwickelte Konzept einer wissensbasierten Konstruktionsunterstützung wurde in Form eines ersten Prototypen realisiert. Die Funktionsfähigkeit des Systems wurde dabei an Beispielen aus der Praxis demonstriert.

MIDAS
A Knowledge-based System for Support of the Fabrication-related Design of LIGA-Microstructures

Abstract

The LIGA-process is a technique which has been developed for the fabrication of microstructures. In the recent years this technology has been used more and more for industrial purposes, from which additional economical requirements arise. The key for an efficient use of farbrication-facilities is a fabrication-related design, because the design of a microstructure may affect the whole following fabrication and mounting process in a positive or a negative manner. Due to this it is important to consider technological influences on a product`s shape already during construction.
For this an appropriate fabrication knowledge is required. However the extensive knowledge needed for the construction of LIGA-microstructures has reached a high level of complexity in the meantime. Therefore the designer depends on an adequate support through information techniques. For this purpose the knowledge-based system MIDAS (Microstructure Diagnosis and Answerbook System) which is the subject of this thesis has been developed.
The first step to a knowledge-based design support is to provide a tool which allows acquiring and accessing fabrication knowledge interactively. Therefore MIDAS contains an information and acquisition component which is based on object-oriented models of the fabrication-knowledge.
The next level of design support is a so-called design rule check. The aim of this procedure is to verify the correctness of a layout. The verification procedure is based on technological constraints concerning a microstructure`s shape and dimensions. For this purpose MIDAS provides a diagnosis tool which performs a design rule check based on an appropriate set of design rules. Hereby the values of critical design features are extracted from the layout and compared with a technologically recommendable value range. Every violation of a design rule is recorded by the tool. Finally the results of the design rule check are displayed.
The concept described above has been realized in a first prototype. The realisability of the concepts could be proved by examples based on real layouts.

Inhaltsverzeichnis

1 Einleitung 1
1.1 Motivation 1
1.2 Ziel der Arbeit 4
1.3 Gliederung der Arbeit 5

2 Das LIGA-Verfahren 7
2.1 Lithographie 7
2.1.1 Maskenherstellung 9
2.1.2 Bestrahlung und Entwicklung 9
2.2 Galvanoformung 11
2.3 Abformung 11
2.3.1 Herstellung eines Abformwerkzeuges 12
2.3.2 Kunststoffabformungsverfahren 13
2.3.3 Herstellung von Metallstrukturen mittels 2. Galvanoformung 13
2.3.4 Herstellung von Keramikstrukturen durch Schlickerguß 14
2.4 Vereinzelung und Montage 15

3 Analyse des für den Entwurf benötigten Fertigungswissens 16
3.1 Klassifizierung technologisch bedingter Einflüsse auf den Entwurf 16
3.1.1 Semantische Klassifizierungskriterien 16
3.1.2 Algorithmische Klassifizierungskriterien 20
3.1.3 Konsistenz und Wertigkeit von Gestaltungsregeln 23
3.2 Anforderungen an die Modellierung des Fertigungswissens 25

4 Die Informationskomponente von MIDAS 26
4.1 Problemstellung bei der Modellierung des Fertigungswissens 26
4.2 Zur Wissensdarstellung verwendete Methode 26
4.3 Objektorientierte Modellierung des Fertigungswissens 27
4.4 Aufbau des Informationssystems 31
4.5 Werkzeug zur Wissensakquisition 33

5 Die Diagnosekomponente von MIDAS 35
5.1 Die Architektur des Diagnosesystems 35
5.1.1 Basiskomponenten eines wissensbasierten Systems 35
5.1.2 Ablauf einer Entwurfsdiagnose 36
5.1.3 Der Aufbau des Diagnosesystems 37
5.2 Vorgabe von Regelwissen 39
5.2.1 Grundelemente einer regelbasierten Wissensdarstellung 39

5.2.2 Darstellung von Entwurfsregeln in Form der Prädikatenlogik 39
5.2.3 Modellierung von Entwurfsrandbedingungen 40
5.2.4 Anbindung von Randbedingungen an ein Regelobjekt..... 41
5.2.5 Erweiterung der Wissenserwerbskomponente 42
5.2.6 Zusammenhänge zwischen Entwurfsregeln 44
5.2.7 Auswahl eines Regelsatzes 44
5.3 Merkmalsgenerierung 45
5.3.1 Merkmale 45
5.3.2 Objektorientierte Modellierung von Merkmalen 46
5.3.3 Merkmalsgeneratoren 48
5.3.4 Abtastverfahren und algebraische geometrische Operationen.. 48
5.3.5 Zellbasierter Ansatz 49
5.3.6 Figurbasierter Ansatz..... 50
5.3.7 Diskussion der verschiedenen Methoden zur Merkmalsgenerierung 57
5.3.8 Aufbau des Moduls zur Merkmalsgenerierung 57
5.4 Inferenz 59
5.4.1 Grundlagen 59
5.4.2 Die angewandte Inferenzstrategie 60
5.5 Interpretation der Diagnoseresultate..... 60
5.5.1 Modellierung von Fehlermeldungen 61
5.5.2 Aufbau des Moduls zur Fehlervisualisierung 63

6 Der Ablauf einer Entwurfsdiagnose am Bsp. eines Planetengetriebes 65
6.1 Der zur Anwendung kommende Regelsatz 65
6.2 Erfassung der Regeln 67
6.3 Durchführung der Entwurfsdiagnose 71
6.3.1 Der zu prüfende Entwurf..... 72
6.3.2 Merkmalsgenerierung 72
6.3.3 Auswahl eines Regelsatzes..... 73
6.3.4 Diagnose..... 74
6.3.5 Darstellung der Diagnoseresultate 76

7 Zusammenfassung und Ausblick 79
7.1 Zusammenfassung 79
7.2 Ausblick..... 80

A Implementierung von MIDAS..... 82

B Das zur Wissensrepräsentation verwendete Datenmodell..... 84
B.1 Repräsentation von Wissensklassen und ihren Instanzen..... 84
B.2 Klassen zur Wissensrepräsentation..... 85

B.2.1 Die Klasse KnowledgeObject 85
B.2.2 Die Klasse KnowledgeObjectScheme 87
B.2.3 Die Klasse Attribute 87
B.2.4 Die Klasse StringAttribute 87
B.2.5 Die Klasse TextAttribute 87
B.2.6 Die Klasse FloatAttribute 87
B.2.7 Die Klasse IntAttribute 88
B.2.8 Die Klasse BitmapAttribute 88
B.2.9 Die Klasse AttributeSetup 88
B.2.10 Die Klasse StringAttributeSetup 88
B.2.11 Die Klasse TextAttributeSetup 88
B.2.12 Die Klasse FloatAttributeSetup 88
B.2.13 Die Klasse IntAttributeSetup 89
B.2.14 Die Klasse BitmapAttributeSetup 89

C Zur Entwurfsdiagnose verwendete Klassen 90
C.1 Erweiterungen des Datenmodells für die Entwurfsdiagnose 90
C.1.1 Die Klasse ParBounds 90
C.1.2 Die Klasse ParBoundsList 92
C.1.3 Die Klasse Priority 92
C.1.4 Die Klasse ParameterType 92
C.1.5 Die Klasse ParameterRange 92
C.1.6 Die Klasse BoolRange 92
C.1.7 Die Klasse IntRange 93
C.1.8 Die Klasse FloatRange 93
C.2 Klassen zur Modellierung von Merkmalen 93
C.2.1 Die Klasse DesignParameter 93
C.2.2 Die Klasse GeoDesignParameter 93
C.2.3 Die Klasse ParameterID 93
C.2.4 Die Klasse GeoParameterID 95
C.2.5 Die Klasse ParameterType 95
C.2.6 Die Klasse ParameterValue 95
C.2.7 Die Klasse BoolParValue 95
C.2.8 Die Klasse IntParValue 95
C.2.9 Die Klasse FloatParValue 95
C.3 Klassen zur Repräsentation von Entwurfsfehlern 95
C.3.1 Die Klasse CheckResult 96
C.3.2 Die Klasse CheckResultSet 96

D Die Benutzeroberfläche von MIDAS 98
D.1 Das Konzept der Benutzeroberfläche 98

D.2 Visualisierungsklassen 99
D.2.1 Die Klasse DisplayForm 99
D.2.2 Die Klasse DisplayItem 99
D.2.3 Die Klasse BoolDisplayItem 101
D.2.4 Die Klasse IntDisplayItem 101
D.2.5 Die Klasse FloatDisplayItem 101
D.2.6 Die Klasse IntIntervalDisplayItem 101
D.2.7 Die Klasse FloatIntervalDisplayItem 102
D.2.8 Die Klasse StringDisplayItem 102
D.2.9 Die Klasse TextDisplayItem 103
D.2.10 Die Klasse BitmapDisplayItem 103
D.2.11 Die Klasse SelectionDisplayItem 103
D.2.12 Die Visualisierungsklassen zur Darstellung von Listen 103

E Die Verwaltung persistenter Daten 105
E.1 Das Persistenzkonzept von MIDAS 105
E.2 Zur Verwaltung persistenter Daten entwickelte Klassen 106
E.2.1 Die Klasse KnowledgeObjectManager 108
E.2.2 Die Klasse SchemeEntry 108
E.2.3 Die Klasse SchemeID2Scheme 108
E.2.4 Die Klasse ObjectEntry 108
E.2.5 Die Klasse FullObjectID2Object 109

F Bei Midas verwendete Dateiformate 110
F.1 Beim Informationssystem verwendete Datenformate 110
F.1.1 Format von Namen 110
F.1.2 Format der Klassenindexdatei 110
F.1.3 Format einer Klassenschemadatei 110
F.1.4 Format von Attributkonfigurationen 111
F.1.5 Format einer Integer-Attributkonfiguration 111
F.1.6 Format einer Objektindexdatei 113
F.1.7 Format der Sicherungsdatei eines Objektes 114
F.2 Im Rahmen des Diagnosesystems verwendete Datenformate 114
F.2.1 Dateischnittstelle des Moduls zur Regelselektion 114
F.2.2 Dateiformat eines Merkmalssatzes 114
F.2.3 Dateiformat eines Fehlerprotokolls 115

Literaturverzeichnis 117

Abbildungsverzeichnis

1 Die wichtigsten Prozeßschritte des LIGA-Verfahrens 8
2 Verfahrensschritte zur Herstellung einer Arbeitsmaske [MB93] 10
3 Prinzip der Herstellung eines Formeinsatzes [MB93] 12
4 Galvanoformung mit einer Angußplatte als Elektrode 14
5 Zusammenhang zwischen Produkttyp, Material und Herstellungsprozeß... 18
6 Bsp. für eine beim Entwurf von Mikrostrukturen zu beachtende Randbedingung 18
7 Beispiele für Gestaltungsregeln 19
8 Beispiele für Gestaltungshinweise 20
9 Bsp. für die fertigungsgerechte Gestaltung von Formeinsätzen 22
10 Bsp. für die fertigungsgerechte Gestaltung von Formeinsätzen 22
11 Bsp. für eine konstruktive Maßnahme zur Unterstützung der Entwicklung ...23
12 Bsp für eine Maßnahme zur Vermeidung von Spannungsrissen 24
13 Bsp. für eine konstruktive Maßnahme zur Unterstützung der Vereinzelung 24
14 Bildung von Klassen 28
15 Klassenstruktur des Fertigungswissens 30
16 Zugriffsebenen des Informationssystems 31
17 Navigationsmechanismus des Informationssystems 32
18 Aufbau des Akquisitionswerkzeuges 33
19 Struktur eines wissensbasierten Systems [HK86] 35
20 Architektur des Diagnosesystems 38
21 Schema einer Regel 39
22 Darstellung einer Abstandsregel 39
23 Objektorientierte Darstellung einer Regel 43
24 Zusammenhang zwischen Fertigungsprozeß und Regelmenge 44
25 Klassen zur Modellierung von Merkmalen 47
26 Grundprinzip des Abtastverfahrens 48
27 Beispiele für Boole'sche Maskenoperationen 49
28 Zellbezogener Entwurfsstil 49
29 Klassenhierarchie des Geometriemodells 52
30 Instanziierung einer Entwurfsgeometrie 53
31 Flächenberechnung 54
32 Algorithmen zur Berechnung der Gesamtfläche eines Entwurfes 56
33 Aufbau des Moduls zur Merkmalsgenerierung 58

34 Kern eines wissensbasierten Systems [Wat86] 59
35 Objektorientierte Darstellung eines Fehlerprotokolles 62
36 Aufbau des Moduls zur Fehlervisualisierung 64
37 Layout eines Planetengetriebes 65
38 Beispiele zu Regel 1 66
39 Beispiele zu Regel 2 66
40 Beispiele zu Regel 3 67
41 Das Startfenster von MIDAS 67
42 Kontrollfenster der Wissenserwerbskomponente 68
43 Bildschirmfenster zur Auswahl einer Wissensklasse 68
44 Darstellung von Regel 1 70
45 Darstellung von Regel 1 als Entwurfsrandbedingung 71
46 Kontrollfenster der Diagnosekomponente 71
47 Beispielentwurf und Ausschnittsvergrößerung 72
48 Berechnung des Merkmalssatzes 73
49 Durchführung der Entwurfsdiagnose 75
50 Kontrolloberfläche der Fehlervisualisierung 76
51 Graphische Anzeige des Diagnoseergebnisses 77
52 Anzeige eines Entwurfsfehlers 78
53 Struktur des Gesamtsystems 83
54 Modellierung des für den Entwurf benötigten Fertigungswissens 84
55 Wissensrepräsentation bei MIDAS 85
56 Klassen zur Wissensrepräsentation 86
57 Erweiterung des Datenmodells für die Entwurfsdiagnose 90
58 Klassen zur Modellierung von Entwurfsrandbedingungen 91
59 Klassen zur Modellierung von Merkmalen 94
60 Klassen zur Darstellung von Entwurfsfehlern 97
61 Beispiel für ein Bildschirmformular 99
62 Visualisierungsklassen 100
63 Persistenzkonzept von MIDAS 106
64 Klassen zur Verwaltung persistenter Daten 107

1. Einleitung

1.1 Motivation

Das im Forschungszentrum Karlsruhe entwickelte LIGA-Verfahren[1] zur Herstellung von Mikrostrukturen [EB90,MB93] befindet sich derzeit auf dem Weg in eine industrielle Nutzung. Charakteristisch für diese Technologie ist eine breite Palette an einsetzbaren Materialien und die Möglichkeit einer nahezu beliebigen lateralen Formgebung. LIGA-Mikrostrukturen zeichnen sich außerdem durch hohe Aspektverhältnisse aus, die mit anderen Mikrostrukturierungstechniken wie Silizium-Technologien nicht erreicht werden.

Entscheidend für die Marktfähigkeit einer Technologie sind neben der Qualität der Produkte aber in erster Linie die zugrundeliegenden Herstellungskosten, sowie die bis zur Marktreife eines Produktes benötigte Entwicklungszeit (time-to-market). Als lithographiebasiertes Verfahren bietet die LIGA-Technik wie die Mikroelektronik die Möglichkeit einer kostengünstigen Massenproduktion durch die parallele Fertigung vieler Einzelkomponenten auf einer Prozeßfläche (Batch-Verfahren). Außerdem können LIGA-Mikrostrukturen durch Abformung schnell und mit geringem Aufwand repliziert werden. Damit beinhaltet die LIGA-Technik ein großes Potential für die industrielle Fertigung miniaturisierter Produkte aus vielen technischen Anwendungsbereichen, jedoch ist der für einen großtechnischen Einsatz der LIGA-Technologie unabdingbare Grad an Automatisierung bislang noch nicht in allen Verfahrensschritten erreicht.

Der Hauptgrund hierfür ist die dynamische Entwicklung der Mikrostrukturtechnik, die häufige Innovationen und damit Änderungen in den Herstellungsverfahren zur Folge hat. Während man im Falle der Mikroelektronik nach einer drei Jahrzehnten langen Entwicklung auf gut erforschte, fertigungstechnisch optimierte Verfahren und standardisierte Baugruppen zurückgreifen kann [Brü93], befindet man sich bei der LIGA-Technik auf vielen Gebieten noch in einem experimentellen Stadium. Im Zuge der laufenden Entwicklungsarbeiten werden immer wieder Prozeßschritte verbessert oder durch neue Alternativen ersetzt, zum Teil erfordert die Realisierung neuer Produkttypen auch die Entwicklung eigener Prozeßvarianten. Komplexität und Umfang des vorhandenen Fertigungswissens sind inzwischen aufgrund der hohen Zahl unterschiedlicher Prozeßschritte und der Vielfalt an Prozeßvarianten ohne eine angemessene informationstechnische Unterstützung nicht mehr handhabbar. Möglichkeiten einer strukturierten Sammlung von Prozeßwissen wurden bereits in [Bra94] untersucht.

Wie die Erfahrungen in der Mikroelektronik zeigen [Brü93], ist bei einer derart aufwendigen Technologie die konsequente Nutzung rechnergestützter Werkzeuge vor allem in der Entwurfsphase unabdingbar, um technologische Einflüsse auf die Pro-

1. Das Akronym LIGA steht für die Hauptschritte des Verfahrens: Lithographie, Galvanoformung und Abformung

duktgestalt bereits bei der Planung von Bauteilen berücksichtigen zu können. Damit ist gewährleistet, daß fertigungstechnisch ungünstige oder im Extremfall nicht realisierbare Entwurfskonstellationen bereits im Vorfeld weitgehend ausgeschlossen werden können und gar nicht erst in die Fertigung gelangen. Man spricht in diesem Zusammenhang von einem *fertigungsgerechten Entwurf.*

Während in der Mikroelektronik die fertigungsgerechte Gestaltung von Schaltungsentwürfen schon seit geraumer Zeit in einer sogenannten Entwurfsregelprüfung (design rule check) sichergestellt wird, befindet sich die Rechnerunterstützung für den Entwurf von Mikrostrukturen noch im Anfangsstadium.

Im Umfeld der LIGA-Technik wurden die sich aus dem fertigungstechnischen Kontext ergebenden Restriktionen für den Entwurf von Mikrostrukturen erstmals in [Leß92] systematisch in Form von Gestaltungsregeln erfaßt. Der direkte Nutzen der Einhaltung dieser Regeln konnte dabei in einem Kostenvergleich nachgewiesen werden. Die Frage inwieweit ein Entwurf fertigungsgerecht durchgeführt wird, hängt jedoch ohne eine angemessene informationstechnische Umsetzung der Gestaltungsregeln in starkem Maße vom individuellen Kenntnisstand und oft auch von der Intuition des Entwerfenden ab.

Ein erster Schritt zu einer Konstruktionsunterstützung besteht zunächst darin, dem Konstrukteur entwurfsbegleitend einen geeigneten interaktiven Zugang zu den jeweils von ihm benötigten Informationen zu Verfügung zu stellen, so daß er mit entsprechendem Hintergrundwissen selbst in der Lage ist, zumindest größere Entwurfsfehler zu vermeiden. Angesichts meist umfangreicher und komplexer Entwürfe und einer Vielzahl zu beachtender Einflußgrößen, sollte der Konstrukteur hiervon jedoch weitgehend durch Werkzeuge für eine automatische Entwurfsprüfung entlastet werden.

Der zunächst naheliegende Schritt, die hierfür in der Mikroelektronik eingesetzten Systeme in entsprechend modifizierter Form zu übernehmen, scheitert prinzipiell an den grundlegenden Unterschieden zwischen den beiden Technologien [Leß92, Tra93, Hahn95]:

- Ein für den Erfolg der Mikroelektronik ausschlaggebender Faktor ist die Verwendung standardisierter Komponenten. Standardbauteile wie Transistoren sind in ihrem physikalischen Verhalten inzwischen hinreichend erforscht und in ihrer Gestaltung optimiert. Die Funktion einer integrierten Schaltung kann auf der Basis einfacher Modelle dieser Bauteile simuliert und damit auch validiert werden.
  Während die Funktionen mikroelektronischer Bauelemente im wesentlichen auf die Steuerung von elektrischen Strömen ausgelegt sind, kommen innerhalb von Mikrosystemen je nach Funktionalität zusätzliche physikalische oder chemische Wirkprinzipien zur Anwendung, die in Hinblick auf die Simulation oder Optimierung von Mikrokomponenten oder Mikrosystemen weitaus komplexere Ansätze erfordern. Aus diesem Grund fehlen der Mikrosystemtechnik bislang mit der

Mikroelektronik vergleichbare Standardbauteile.

- Bedingt durch die Dreidimensionalität von Mikrostrukturen und der Möglichkeit, diese in nahezu beliebigen lateralen Formen zu fertigen, sind die zur Überprüfung von VLSI-Maskenlayouts eingesetzten polygonorientierten Algorithmen [Brü93] im Bereich der LIGA-Technik nur bedingt anwendbar. In der Regel basieren mikroelektronische Layouts auf rechtwinkligen (Manhattan style) oder oktogonalen Strukturen mit 45°-Winkeln. Während diese Vereinfachung in der Mikroelektronik keine signifikante Einschränkung der elektrischen Funktionalität zur Folge hat, sind die meisten Anwendungen der Mikrotechnik mit derart einfachen geometrischen Formen nicht realisierbar. Oftmals erfordern Probleme aus Anwendungsbereichen wie der Mikrooptik oder Mikrofluidik komplizierte Konstruktionen, die mit Polygonen auch bei Verwendung einer Vielzahl von Stützstellen nur unzureichend beschrieben werden können.
  Wegen der diesbezüglich besseren Darstellungsmöglichkeiten greifen LIGA-Konstrukteure auch häufig auf CAD-Systeme für den Mechanikentwurf zurück, die allerdings auf einem anderen Datenformat, meist IGES [IGES91], aufsetzen als Layouteditoren der Mikroelektronik, die in der Regel auf GDS II [Cal84] oder CIF basieren.
- Im Gegensatz zur Mikroelektronik verfügt man bei der LIGA-Technik über keine festen Fertigungssequenzen. Der Prozeßverlauf wird in der Regel an den jeweils zu fertigenden Produkttyp in Abhängigkeit vom vorgesehenen Arbeitsmaterial und der Grundgeometrie angepaßt. Dementsprechend muß auch der bei einer Entwurfsdiagnose anzuwendende Regelsatz individuell für jedes Produkt festgelegt werden. Fertigungsplanung und Entwurf erfolgen üblicherweise weitgehend parallel und iterativ bis Funktionalität, Design und Fertigungsprozeß aufeinander abgestimmt sind.
- Aufgrund des unterschiedlichen physikalisch-technischen Hintergrundes der einzelnen Prozeßschritte des LIGA-Verfahrens sind die sich aus ihnen ergebenden Gestaltungsregeln untereinander nicht notwendigerweise konsistent.
  Hinzu kommt, daß viele der z.B. in [Leß92] aufgeführten Regeln oft nur empfehlenden oder rein qualitativen Charakter besitzen und vom Konstrukteur beim Testen neuer Varianten zum Teil auch bewußt außer acht gelassen werden. Die Entscheidung, welche Regeln bei einem Entwurf zur Anwendung kommen oder wie Konsistenzkonflikte innerhalb eines Regelsatzes aufgelöst werden, obliegt in letzter Konsequenz dem Konstrukteur. Dieser ist somit weitaus stärker in die Entwurfsdiagnose involviert als dies beim VLSI-Entwurf der Fall ist.

Aus den genannten Punkten ergibt sich die Notwendigkeit, für die Konstruktion von LIGA-Mikrostrukturen dedizierte Werkzeuge [FFL94, SEBSW94, BKS95, Hahn95] bereitzustellen, die den speziellen Anforderungen der komplexen LIGA-Technologie gerecht werden. Es besteht hierbei zum einen Bedarf an einem Informationssystem, das dem Konstrukteur jeweils das in den verschiedenen Entwurfsphasen benötigte

Fertigungswissen bereitstellt, zum anderen werden Werkzeuge benötigt, die dem Konstrukteur fertigungstechnisch kritische Elemente einer Konstruktionszeichnung aufzeigen.

Einen Ansatz hierzu stellt das im Laufe dieser Arbeit entstandene wissensbasierte System MIDAS (Microstructure Diagnosis and Answerbook System) dar.

1.2 Ziel der Arbeit

Ziel dieser Arbeit ist es, dem Konstrukteur von LIGA-Mikrostrukturen ein informationstechnisches Werkzeug zur Unterstützung eines fertigungsgerechten Entwurfes an die Hand zu geben. Entsprechend der im vorigen Abschnitt genannten Ansatzpunkte für eine derartige Konstruktionsunterstützung handelt es sich bei den grundlegenden Aufgaben des Systems um die Darstellung von entwurfsrelevantem Fertigungswissen, sowie eine auf diesem Wissen basierende Diagnose von Mikrostrukturentwürfen.

Aufgrund der fortschreitenden Entwicklung der LIGA-Technologie unterliegt dieses Wissen jedoch häufigen Änderungen. Außer den genannten Systemkomponenten müssen daher auch geeignete Werkzeuge zur Wissensakquisition und Aktualisierung der Wissensbasis bereitgestellt werden.

Die Informationskomponente von MIDAS

Der erste Schritt bei der Entwicklung der Informationskomponente von MIDAS ist eine Analyse des dafür benötigten Fertigungswissens mit dem Ziel einer Modellbildung, die sich zum einen an den spezifischen Anforderungen dieser Anwendung orientiert, zum anderen aber auch in Richtung einer Anwendung im Rahmen einer Entwurfsdiagnose und der Wissenserfassung erweiterbar ist.

Die Aufgabe der Informationskomponente des Systems besteht darin, dem Anwender das erfaßte Fertigungswissen in einer seinem Informationsbedarf angemessenen Weise zugänglich zu machen, d.h. die Darstellung des Wissens muß sich an der Begriffswelt, bzw. Semantik des zu repräsentierenden Wissensbereiches orientieren. Es ist daher sinnvoll das Fertigungswissen entsprechend dieser semantischen Kriterien zu klassifizieren. Am besten eignen sich für eine derartige Problemstellung objektorientierte Ansätze [ThSc89], da hierbei die semantisch grundlegenden Begriffe des Anwendungsgebietes als Klassen in das Datenmodell übertragen werden können [Bra94]. Ein Beispiel hierfür ist im vorliegenden Fall die Einführung einer Klasse von Gestaltungsregeln.

Entsprechend der wesentlichen Funktion eines Informationssystems, der Darstellung von Wissen, besteht die Hauptaufgabe der diesen Klassen angehörenden Objekte darin, das in ihnen enthaltene Fertigungswissen zu visualisieren. Zur Erfassung dieses Wissens sind außerdem Mechanismen zur Eingabe und die persistente Speicherung von Informationen bereitzustellen.

Das eigentliche Programm besteht bei der Informationskomponente und dem dazugehörigen Eingabewerkzeug im wesentlichen aus einer übergeordneten Benutzeroberfläche, die auf diese Mechanismen zurückgreift.

Die Diagnosekomponente von MIDAS

Mit Hilfe der Diagnosekomponente des Systems sollen Produktentwürfe auf der Grundlage eines geeignet definierten Regelsatzes bezüglich ihrer fertigungstechnische Eignung untersucht werden. Für den Fall eines Entwurfsfehlers ist die Ausgabe einer entsprechenden Meldung mit bezug auf das betroffene Entwurfsmerkmal und die verletzte Randbedingung vorgesehen.

Während die Arbeitsweise der Informationskomponente im wesentlichen durch die Struktur des darzustellenden Fertigungswissens bestimmt ist, basiert die Funktionsweise der Diagnosekomponente von MIDAS auf der Abfolge einzelner Diagnoseschritte. Bei der Entwicklung dieser Systemkomponente ist es daher sinnvoll, zunächst den Ablauf einer Entwurfsdiagnose unter dem Aspekt einer funktionalen Dekomposition zu betrachten, um auf deren Basis die grundlegende modulare Struktur des Diagnosesubsystems festzulegen und davon ausgehend eine Erweiterung der vorhandenen Klassenstruktur vorzunehmen.

Die wesentlichen Funktionen der Diagnosekomponente sind:

- die Vorgabe, bzw. Erstellung des bei der Diagnose anzuwendenden Regelsatzes
- die Extraktion der für die Diagnose relevanten Entwurfsparameter aus der Konstruktionszeichnung
- die Überprüfung der ermittelten Entwurfsparameter auf der Grundlage des vorgegebenen Regelsatzes
- die Darstellung und Interpretation der Diagnoseresultate im Dialog mit dem Konstrukteur

Aus den Aufgaben der entsprechenden Subsysteme ergeben sich zusätzliche Anforderungen an das im Rahmen des Informationssystems entwickelte Datenmodell. Einerseits muß die vorhandene objektorientierte Datenstruktur zur Erfassung von diagnoserelevantem Regelwissen erweitert werden, andererseits werden zusätzliche für die Durchführung der einzelnen Diagnoseschritte und den Datenaustausch zwischen den verschiedenen Subsystemen verantwortliche Klassen benötigt.

Am Schluß der Arbeiten steht die Integration der einzelnen Module in eine übergeordnete Kommandostruktur.

1.3 Gliederung der Arbeit

Im nachfolgenden Kapitel wird zunächst ein Überblick über das LIGA-Verfahren gegeben.

Darauf aufbauend befaßt sich das dritte Kapitel mit dem Aspekt fertigungsbedingter Einflüsse auf die Produktgestalt. In einer detaillierten Analyse wird dabei erörtert, welche Art von Fertigungswissen für den Konstrukteur in den unterschiedlichen Entwurfsphasen von Relevanz ist, und wie dieses Wissen im Rahmen einer Konstruktionsunterstützung eingesetzt werden kann.

Gegenstand des vierten Kapitels ist das Informationssystem. Auf der Basis eines objektorientierten Modells des Fertigungswissens werden in diesem Abschnitt Architektur und Funktionalität des Informationssystems beschrieben. Der abschließende Abschnitt dieses Kapitels befaßt sich mit dem Aspekt der Wissensakquisition.

Im fünften Kapitel werden Aufbau und Funktionsweise des Diagnosesystems vorgestellt. Ausgehend von einer Analyse des Diagnoseablaufs behandelt dieses Kapitel die einzelnen Subsysteme dieser Komponente und die dafür entwickelten Datenmodelle.

Mit der Anwendung des Systems befaßt sich das sechste Kapitel. Darin wird der Ablauf einer Entwurfsdiagnose in allen Schritten - von der Vorgabe der Entwurfsregeln bis hin zur Interpretation des Diagnoseresultats - anhand eines Anwendungsbeispiels beschrieben.

Den Abschluß der Arbeit bilden eine Zusammenfassung und ein kurzer Ausblick auf eine mögliche Weiterentwicklung des Systems.

Auf Aspekte der Realisierung des Systems und dessen Implementierung gehen die Anhänge ein.

2. Das LIGA-Verfahren

Für ein besseres Verständnis der dem Entwurf von LIGA-Mikrostrukturen zugrundeliegenden Problematik soll in diesem Kapitel zunächst ein Überblick über den Ablauf des LIGA-Prozesses gegeben werden [Leß92,MB93,Hahn95].

Die Anfänge der LIGA-Technologie reichen bis zum Ende der 70er Jahre zurück [Beck82]. Ziel der damals am Kernforschungszentrum Karlsruhe begonnenen Entwicklungsarbeiten war die kostengünstige Herstellung von Trenndüsen, die in der Kerntechnik bei der Uran-Isotopentrennung zur Anwendung kommen. Mittlerweile wird das Verfahren zur Fertigung von Mikrokomponenten für die verschiedensten Anwendungsbereiche eingesetzt.

Obwohl inzwischen eine Vielzahl unterschiedlicher Prozeßvarianten entstanden ist, setzt sich der LIGA-Prozeß im wesentlichen aus vier grundlegenden Verfahren zusammen, die entweder dem lokalen Aufbau oder Entfernen von Material dienen [Leß92]. Im einzelnen handelt es sich hierbei um:

- Lithographie
- selektives, naßchemisches Ätzen von Opferschichten
- Galvanoformung
- Formgebung mit Hilfe von Mutterformen

Diese Grundprozesse werden in verschiedener Weise dazu genutzt, um mit Hilfe einer Kette von Zwischenprodukten das gewünschte Endprodukt zu gewinnen. Einen Überblick über den typischen Ablauf des LIGA-Verfahrens gibt Abb.1. Die einzelnen Verfahrensschritte sollen in den folgenden Abschnitten vorgestellt werden.

2.1 Lithographie

Ausgangspunkt des Verfahrens ist eine mit Hilfe eines CAD-Systems erstellte Konstruktionszeichnung. Diese in einem der üblichen Layoutformate, wie z.B. GDSII, vorliegenden Geometriedaten dienen als Grundlage für die Herstellung einer Arbeitsmaske.

Bei der Lithographie überträgt man das Bild dieser Arbeitsmaske entweder durch Projektion oder nach dem Schattenwurfprinzip in eine Resistschicht. In den belichteten Bereichen des Resists wird eine Veränderung in den Molekülketten des Materials hervorgerufen, die man bei der Entwicklung dazu nutzt, durch selektives, naßchemisches Ätzen entweder die belichteten oder die unbelichteten Resistteile zu lösen.

Herkömmliche lichtoptische Lithographiesysteme können dabei lediglich zur Herstellung flacher Strukturen eingesetzt werden, da sie nur über eine geringe Tiefenschärfe verfügen. Bei der Tiefenlithographie greift man daher auf energiereichere Röntgenstrahlung zurück, mit deren Hilfe auch eine Strukturierung extrem dicker Resistschichten möglich ist. Eine hierfür geeignete, hochintensive und nahezu paral-

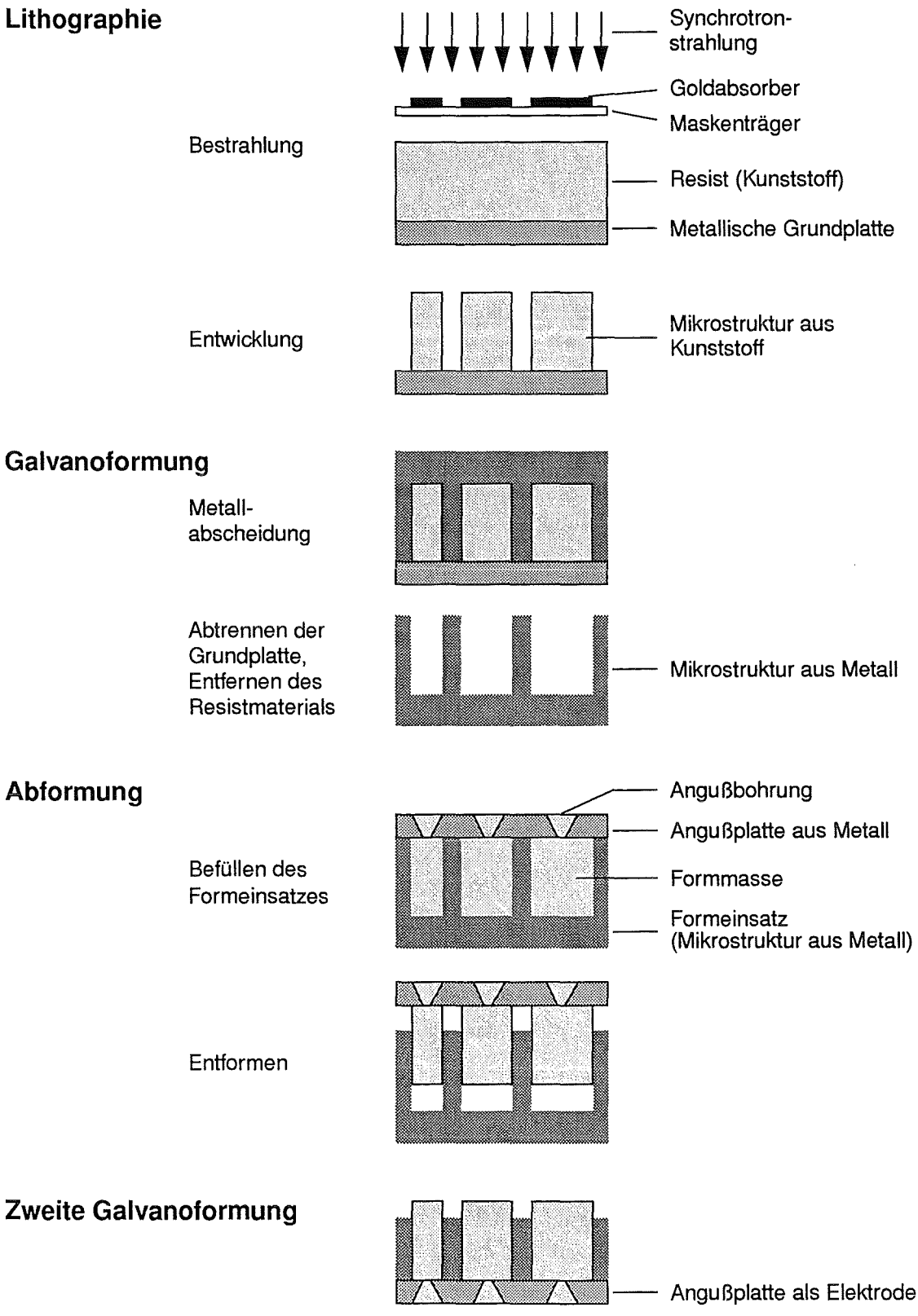


Abbildung 1: *Die wichtigsten Prozeßschritte des LIGA-Verfahrens*

lele Strahlung findet man in der sogenannten Synchrotronstrahlung, die z.B. mit Hilfe eines Elektronenspeicherringes oder eines Elektronensynchrotrons erzeugt werden kann. Die charakteristische Wellenlänge dieser Strahlung liegt typischerweise im Bereich von 0,2 bis 0,6nm.

2.1.1 Maskenherstellung

Die zur Strukturierung des Resistmaterials eingesetze Arbeitsmaske muß in ihren Materialeigenschaften an diese energiereiche Röntgenstrahlung angepaßt sein. Die im sichtbaren Bereich verwendeten Chrommasken sind für diese kurzwellige Strahlung jedoch in hohem Maße transparent. Zur Abschattung von Resistbereichen greift man daher auf Materialien mit hohem Atomgewicht wie z.B. Gold zurück, die ein hohes Absorptionsvermögen für die verwendete Röntgenstrahlung besitzen. Um eine ausreichend starke Absorption zu bewirken, müssen Goldabsorber eine Dicke von etwa 10µm besitzen.

Als Träger der Absorberstrukturen fungiert eine dünne, über einem Rahmen gespannte Membran, die aus einem Material mit möglichst geringem Absorptionsvermögen für die verwendete Röntgenstrahlung besteht. Üblicherweise verwendet man hierfür Trägerfolien aus Titan oder Beryllium.

Den Ablauf der Maskenherstellung zeigt Abb.2. Zunächst wird eine etwa 2µm dicke Titanschicht auf einen Träger aus Invar aufgesputtert und darauf eine Resistschicht aufgebracht. Anschließend strukturiert man diese mittels Synchrotronstrahlung. Dabei greift man in der Regel auf eine sogenannte Zwischenmaske von etwa 3µm Höhe zurück, die entweder durch optische Kopie einer herkömmlichen Chrommaske oder direkte Elektronestrahllithographie [Hei92] gewonnen wird. Dieser Schritt ist erforderlich, da eine hinreichend präsize Strukturierung der zur Herstellung einer Arbeitsmaske erforderlichen Resistschichten derzeit nur mit Synchrotronstrahlung möglich ist.

Die bei der Strukturierung des Resists entstandenen Zwischenräume werden anschließend galvanisch mit Gold gefüllt und das verbliebene Resistmaterial chemisch entfernt. Danach wird der Invarträger in einem Fensterbereich von der Rückseite her weggeätzt, so daß dort eine freitragende Titanschicht übrigbleibt.

2.1.2 Bestrahlung und Entwicklung

Die Arbeitsmaske wird dazu verwendet, die vorgegebene Geometrie auf ein mit Resistmaterial beschichtetes Substrat zu übertragen. Die Bestrahlung bewirkt einen Abbau der Molekülketten des Resistmaterials, was eine Änderung seiner chemischen Eigenschaften zur Folge hat.

Die unterschiedlichen chemischen Eigenschaften von bestrahltem und unbestrahltem Resistmaterial nutzt man bei der Entwicklung dazu, durch selektives, naßchemisches Ätzen entweder die belichteten (Positivresist) oder die unbelichteten (Negativresist) Resistteile zu lösen.

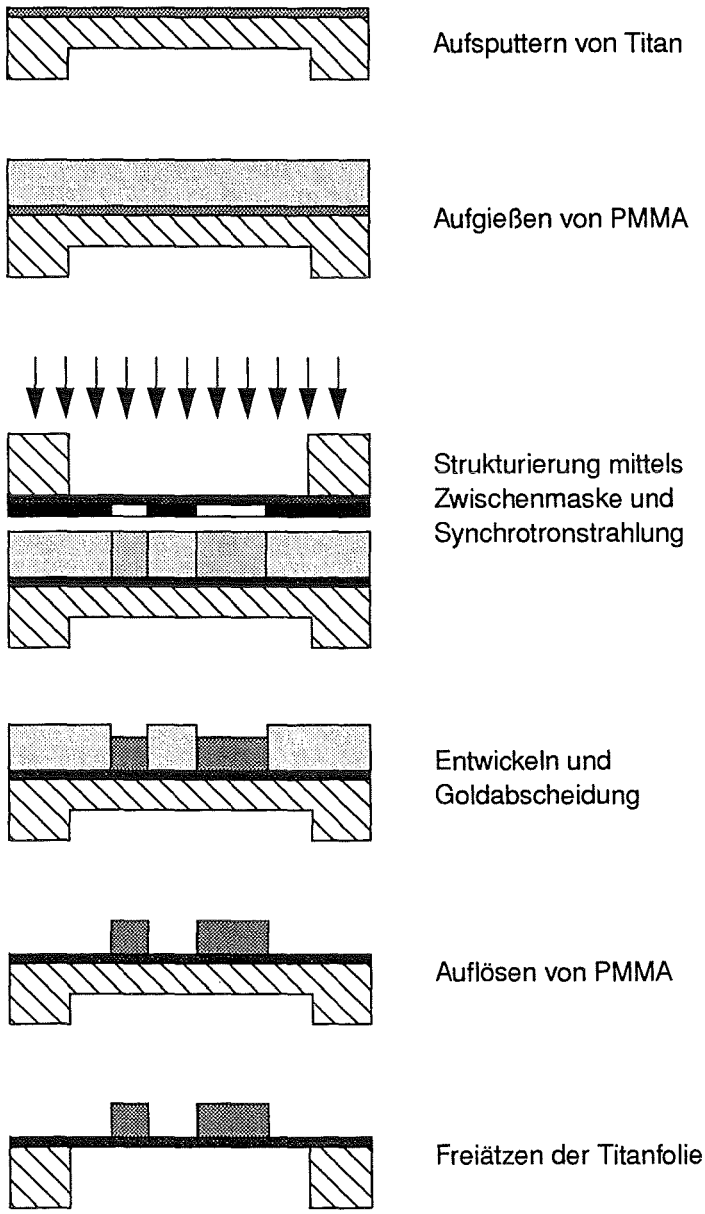


Abbildung 2: *Verfahrensschritte zur Herstellung einer Arbeitsmaske [MB93]*

In der Regel verwendet man als Resistmaterial Polymethylmethacrylat (PMMA), gemeinhin bekannt als Plexiglas. Mit Hilfe eines speziellen Gießprozesses [Mohr88] können Resistschichten von 10µm bis zu mehreren hundert µm Dicke hergestellt werden.

Die chemischen Eigenschaften dieses Materials ermöglichen eine feine Strukturierung und damit eine hohe Fertigungsgenauigkeit. Ein besonders für die spätere Abformung wichtiger Faktor ist die geringe Kantenrauhigkeit (<30nm) von PMMA. Hierdurch ist es möglich, die Seitenwände von Mikrostrukturen mit hoher Güte zu fertigen.

Die nach der Entwicklung auf dem Substrat verbliebenen Kunststoffkörper (Primärstrukturen) können bereits als Endprodukte verwertet werden. Im allgemeinen verwendet man Primärstrukturen jedoch als Ausgangsbasis weiterer Herstellungsschritte.

2.2 Galvanoformung

Ziel der Galvanoformung ist die Herstellung metallischer Mikrostrukturen. Diese werden dadurch gewonnen, daß man ein Substrat mit einer strukturierten Resistschicht in eine Elektrolytlösung bringt und zwischen Substrat und einer Referenzelektrode eine Gleichspannung anlegt. Das elektrisch leitfähige Material des Substrats dient dabei als Kathode. überall, wo die Substratoberfläche freigelegt wurde, lagern sich die positiv geladenen Metallionen des Elektrolyten ab und füllen so nach und nach die Resistzwischenräume. Je nach Verwendungszweck läßt man das Metall bis zur Oberkante der Resiststrukturen oder darüber hinaus wachsen. Das verbliebene Resistmaterial kann anschließend durch Ätzprozesse herausgelöst werden.

Die Mikrogalvanik ist bei verschiedenen Fertigungsschritten des LIGA-Verfahrens von zentraler Bedeutung. Wichtigste Anwendung der Galvanoformung ist die Herstellung von Abformungswerkzeugen, die eine schnelle und kostengünstige Replikation von Mikrostrukturen ermöglichen. Daneben wird die Mikrogalvanik, wie bereits erwähnt, bei der Maskenherstellung zur Fertigung der Absorberstrukturen eingesetzt.

2.3 Abformung

Bei der geschilderten Herstellung sogenannter Direkt-LIGA-Strukturen, also Metallstrukturen die durch Röntgentiefenlithographie und Galvanoformung gefertigt werden, handelt es sich um einen extrem aufwendigen Prozeß. Für jedes Substrat muß eine erneute Strukturierung mit allen Arbeitsschritten - vom Resistauftrag bis zur Entwicklung - durchgeführt werden. Eine weitaus kostengünstigere Alternative hierzu stellt die Abformung mit Kunststoffen dar.

2.3.1 Herstellung eines Abformwerkzeuges

Da die bei der Abformung verwendeten Mutterformen zum Teil erheblichen mechanischen und thermischen Belastungen ausgesetzt sind, verwendet man dafür Abformwerkzeuge aus Metall, die über eine hinreichende Formstabilität verfügen.

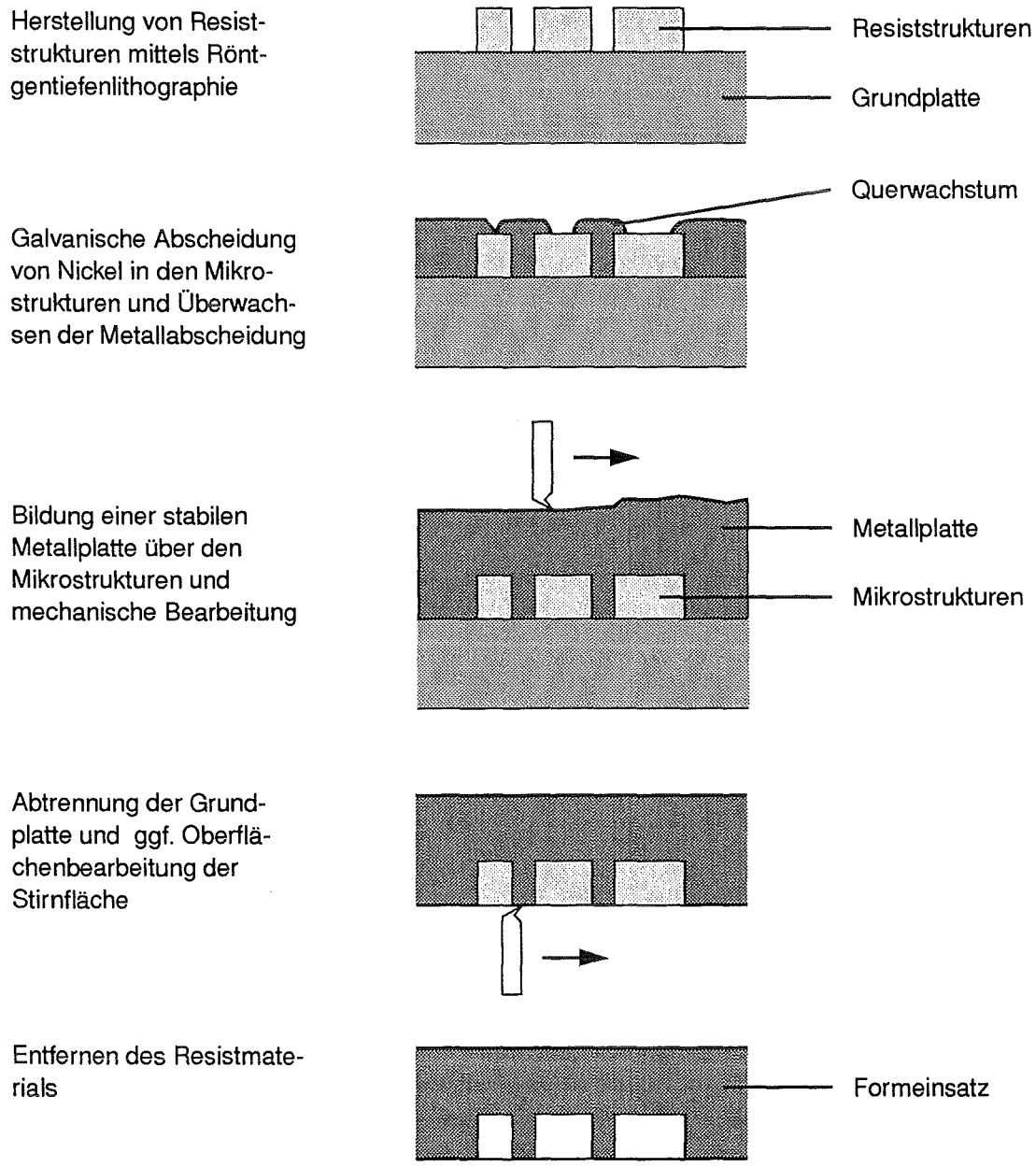


Abbildung 3: *Prinzip der Herstellung eines Formeinsatzes [MB93]*

Die Herstellung dieser Werkzeuge erfolgt mittels Galvanoformung (siehe Abb.3). Dabei läßt man das Metall über die Resiststrukturen hinauswachsen, bis sich über diesen eine stabile, etwa 5mm dicke Metallplatte gebildet hat. Nach einer Oberflächenbearbeitung der Formrückseite wird der Formeinsatz von der Grundplatte entfernt und das verbliebene Resistmaterial nach einer gegebenenfalls notwendigen Glättung der Stirnfläche gelöst. Die so gefertigten Formeinsätze können problemlos Temperaturen von etwa 150°C und Drücken bis zu10 MPa ausgesetzt werden.

2.3.2 Kunststoffabformungsverfahren

In der LIGA-Technologie kommen verschiedene Kunststoffabformungsverfahren zur Anwendung [MB93]. Man unterscheidet hierbei zwischen Spritzgießverfahren und Prägeverfahren.

Übliche Gießverfahren sind der thermoplastische Spritzguß (TIM = Thermoplastic Injection Moulding) und das Reaktionsharzgießen (RIM = Reaction Injection Moulding).

Beim thermoplastischen Spritzguß werden zunächst die in fester Form - meist als Granulat - vorliegenden Ausgangsstoffe aufgeschmolzen, so daß sie ein mehr oder weniger zähflüssiges Medium bilden. Daraufhin spritzt man die auf diese Weise plastisch gewordene Formmasse unter Druck in die Gußform, wo sie dann bei der Auskühlung wieder erstarrt.

Im Unterschied zum thermoplastischen Spritzguß verwendet man beim Reaktionsharzgießen zwei oder mehr flüssige Reaktanten als Formmasse. Diese werden in einer Mischkammer unter hohem Druck miteinander vermengt und das dabei entstandene Gemisch anschließend in die Gußform geleitet. Dort härtet es durch Polymerisation aus.

Eine weitere im Bereich der LIGA-Technik angewandte Abformungsmethode ist das Prägeverfahren. Ein bevorzugter Anwendungsbereich dieser Methode ist die Abformung von Mikrostrukturen auf bereits prozessierten Silizium-Wafern, d.h. über den mikroelektronischen Schaltungen. Hierbei wird zunächst die Formmasse auf einen mit Schutz- und Metallisierungsschichten versehenen Wafer aufgetragen und nach dem Aushärten auf eine Temperatur, die über der Glasübergangstemperatur des verwendeten Kunststoffes liegt, erwärmt. Die Formgebung erfolgt durch Aufprägen eines evakuierten Formeinsatzes, der nach ausreichender Abkühlung des Kunststoffes wieder entfernt wird.

2.3.3 Herstellung von Metallstrukturen mittels zweiter Galvanoformung

Bei den durch Abformung gewonnenen Kunststoffstrukturen handelt es sich um Replikate der mittels Röntgentiefenlithographie hergestellten Primärstrukturen. Diese replizierten Strukturen können selbst wiederum als Formgebungshilfen zur Herstellung von Metall- oder Keramikstrukturen verwendet werden [MB93, Leß92].

Die Fertigung von Metallstrukturen erfolgt in einer zweiten Galvanoformung. Grundlegende Voraussetzung hierfür ist die Galvanisierbarkeit der Kunststoffstrukturen, d.h. man benötigt eine elektrisch leitfähige Grundplatte, um die Galvanisierung in Gang setzen zu können.

Häufig verwendet man zu diesem Zweck bei der Abformung eine metallische Angußplatte, deren Angußbohrungen durch so ausgeführt sind, daß die beim Befüllvorgang übertretende Formmasse eine formschlüssige Verbindung von Kunststoffstrukturen und Angußplatte ergibt. Bei der Galvanisierung fungiert diese dann als Kathode (siehe Abb.4).

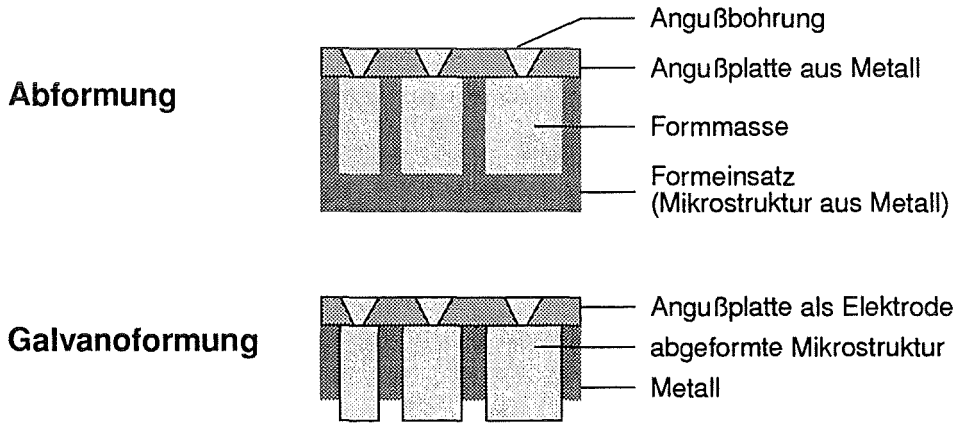


Abbildung 4: *Galvanoformung mit einer Angußplatte als Elektrode*

In anderen Fertigungsvarianten werden zunächst in einem ersten Abformungsschritt die Formnester der Formeinsätze mit elektrisch nicht leitfähigem Kunststoff gefüllt. Nach dem Aushärten der Formmasse überschichtet man diese mit einem elektrisch leitfähigen Material und erhält so die zur Galvanisierung benötigte Trägerplatte.

2.3.4 Herstellung von Keramikstrukturen durch Schlickerguß

Neben verschiedenen Kunststoffen und Metallen können beim LIGA-Verfahren auch keramische Werkstoffe als Abformmaterialien verwendet werden. Die Herstellung von Keramikstrukturen erfolgt im sogenannten Schlickergußverfahren.

Als Abformungswerkzeuge dienen hierbei Kunststoffkörper mit einseitig offenen Kavitäten. Die Formmasse besteht aus einem feinkristallinen Keramikpulver, das zu einem Schlicker eingeschlämmt wird. Mit diesem befüllt man die Kunststofform und verdichtet den Schlicker durch Entzug der Schlämmflüssigkeit. Danach wird der so entstandene Verbund im Ofen gebrannt, wobei sich die Kunststofform zersetzt und eine feste Struktur aus Keramik verbleibt.

2.4 Vereinzelung und Montage

Der eigentlichen Fertigung folgen als abschließende Arbeitsschritte die Vereinzelung der hergestellten Mikrokomponenten und je nach Anwendungszweck ihre Montage zu komplexeren Strukturen oder Systemen. Welche Arbeiten hierbei durchzuführen führen sind, hängt in starkem Maße vom zu fertigenden Produkt ab. Vereinzelung und Montage konnten daher bislang nur in einem geringen Umfang automatisiert werden, d.h. zu einem hohen Anteil bestehen diese Fertigungschritte aus manueller Arbeit.

3. Analyse des für den Entwurf benötigten Fertigungswissens

Ziel eines fertigungsgerechten Entwurfes ist eine effiziente Nutzung der Fertigungspotentiale durch eine in fertigungstechnischer Hinsicht optimierte Gestaltung der Produktentwürfe, d.h. die Geometrie eines Produktes sollte, soweit von den funktionalen Anforderungen her möglich, an das Fertigungsverfahren angepaßt werden. Dies umfaßt zum einen das Ausschließen prozeßtechnisch ungünstiger oder nicht realisierbarer Konstellationen, zum anderen unterstützende Maßnahmen wie beispielsweise das Ausgleichen fertigungsbedingter Maßabweichungen durch einen entsprechenden Maßvorhalt.

Die Voraussetzung für einen derartigen Entwurfsstil ist die Kenntnis technologischer Einflüsse auf die Produktgestalt, insbesondere der sich aus dem Herstellungsverfahren ergebenden Randbedingungen für bestimmte Entwurfsgrößen.

Gegenstand dieses Kapitels ist eine Analyse des für einen fertigungsgerechten Entwurf von LIGA-Mikrostrukturen benötigten Fertigungswissens.

3.1 Klassifizierung technologisch bedingter Einflüsse auf den Entwurf

Nachdem im vorangegangenen Kapitel die wesentlichen Verfahrensschritte des LIGA-Prozesses vorgestellt wurden, sollen im folgenden die gegenseitigen Abhängigkeiten von Produktgestalt und Fertigungstechnologie erörtert werden. Gegenstand dieser Untersuchung sind dabei weniger die physikalisch-technischen Aspekte des Problemkreises - diese werden in [Leß92] ausführlich diskutiert -, viel mehr interessiert in diesem Zusammenhang die Möglichkeit einer Klassifizierung der verschiedenen Einflußgrößen nach informationstechnischen Gesichtspunkten wie semantischen oder algorithmischen Kriterien.

3.1.1 Semantische Klassifizierungskriterien

Eine Klassifizierung des Fertigungswissens nach semantischen Kriterien hat den Zweck, das Wissen in einer Form zu erfassen, die mit der Sichtweise des Anwenders übereinstimmt. Eine derartige Vorgehensweise ist vor allem unter dem Aspekt des Informationsaustausches zwischen Mensch und Maschine - wie er im Rahmen eines Informationssystems gegeben ist - sinnvoll.

Der Informationsbedarf eines Konstrukteurs von LIGA-Mikrostrukturen ist in erster Linie vom Ablauf des Konstruktionsverfahrens geprägt. Hierbei steht aber weniger der physikalisch-technische Aspekt im Vordergrund, der Konstrukteur ist in diesem Zusammenhang viel mehr an den Konsequenzen interessiert, die sich aus dem fertigungstechnischen Hintergrund der einzelnen Verfahrensschritte für die Gestaltung seines Produktes ergeben.

Der Charakter der bei der Konstruktion benötigten Informationen hängt in starkem Maße vom gerade durchlaufenen Stadium des Konstruktionsprozesses ab. In der Anfangsphase sind zunächst Fragen wie die Auswahl einer zur Funktionserfüllung

geeigneten Produktgeometrie, bzw. geeigneten Materials zu klären. Der Konstrukteur sollte hierfür die grundlegenden Realisierungsmöglichkeiten des Verfahrens wie den Grenzen der Produktabmessungen und Fehlertoleranzen kennen (siehe Tabelle 1), und er muß auch in den Grundzügen mit den gängigen Produktvarianten vertraut sein (Beispiel siehe Abb.5).

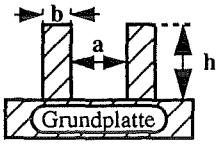
 Mikrostruktur	**Resist-strukturen**	**Metall-strukturen**	**Kunststoff-strukturen**	**Keramik-strukturen**
$(h/b)_{max}$ gestützt	50 - 100	50 - 100	50 - 100	4
$(h/b)_{max}$ freistehend	15	15	15	0-4
h_{max} [µm]	600 ± 5	600 ± 5	600 ± 5	200 - 300
b_{min} [µm]	2 ± 0,5	5 ± 0,5	ca. 5 ± 0,8%	ca. 10 - 20
a_{min} [µm]	5 ± 0,5	2 ± 0,5	ca. 5 ± 0,8%	ca. 10 - 20
Prozeßfläche [mm^2]	20 x 60	20 x 60	20 x 60	ca. 15 x 40
Grundplatte [mm^3]	54 x 84 x 8	54 x 84 x 8	26 x 66 x 2-6	ca. 20 x 45 x 1-2

Tabelle 1: Anhaltswerte für Formate, Maße und Fertigungstoleranzen [Leß92]

Mit der Auswahl eines Materials und eines Produkttypes liegt der zur Anwendung kommende Fertigungsprozeß zumindest in den grundlegenden Verfahrensschritten fest.

Danach steht der Konstrukteur vor der Aufgabe, die Einflüsse der gewählten Herstellungsvariante auf die Produktgestalt abzuschätzen und beim Entwurf zu berücksichtigen. Hierbei ist er im wesentlichen an konkreten Richtlinien, die sich aus den verschiedenen Verfahrensschritten für die Entwurfsgestaltung ergeben, interessiert. Ein Beispiel hierfür zeigt Abb.6

Bei der Vermittlung fertigungstechnischer Randbedingungen für die Gestaltung eines Produktes greift man üblicherweise auf sogenannte *Gestaltungsregeln* zurück [PaBe86, Bode88]. Der Begriff „Gestalt" bezieht sich dabei nicht nur auf die geometrische Form eines Produktes, sondern umfaßt auch nicht-geometrische Eigenschaften wie z.B. Materialkenndaten.

Beispiel: Senkrechte Extrusion einer beliebigen Flächengeometrie, Material Kunststoff,

Bei der senkrechten Extrusion von Flächengeomerie können beliebige laterale Formen vorgegeben werden. Bei der Herstellung von Kunststoffstrukturen kommt folgender Prozeß zur Anwendung:

- *Röntgentiefenlithographie (RTL) von Resisterhebungen*
- *Herstellung eines Formeinsatzes durch Galvanoformung*
- *Abformung*

Der Prozeß hat inzwischen Serienreife erlangt.

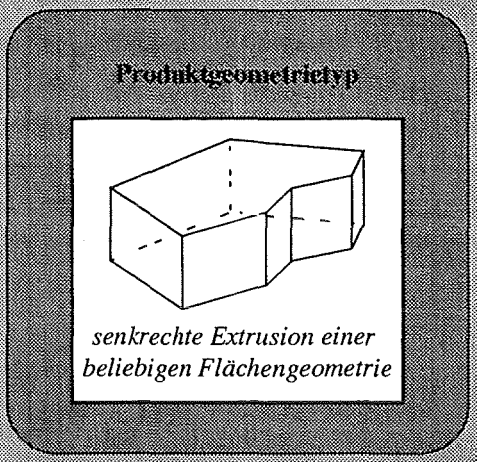


Abbildung 5: *Zusammenhang zwischen Produkttyp, Material und Herstellungsprozeß*

Beispiel: Maskentechnik

Ein Problem bei der Bestrahlung eines Substrates ist die unkontrollierte Reflexion von Röntgenstrahlung an den Seitenwänden der Absorberstrukturen

Verursacht wird dies zum einen durch die natürliche Divergenz der Synchrotronstrahlung, zum anderen durch zu rauhe Absorberseitenwände. Hierdurch kann es zur Belichtung abgeschatteter Resistbereiche kommen, falls die Absorberstrukturen dichter als etwa 2µm zusammenliegen. Beim Entwurf ist daher auf die Einhaltung eines entsprechenden Mindestabstandes zwischen den Strukturen zu achten.

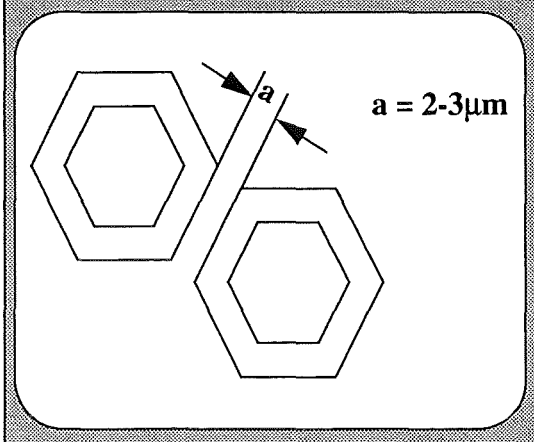


Abbildung 6: *Beispiel für eine beim Entwurf von Mikrostrukturen zu beachtende Randbedingung*

Der wichtigste Bestandteil einer Gestaltungsregel ist die Regelformulierung. Diese schildert knapp die für die Produktgestaltung relevanten Fakten. Daneben beinhalten Gestaltungsregeln außerdem eine kurze Begründung, die den physikalisch-technischen Hintergrund umreißen soll, und graphische Beispiele für fertigungstechnisch günstige oder nicht empfehlenswerte Entwurfskonstellationen (siehe Abb.7).

Die beim Entwurf von LIGA-Strukturen zu beachtenden Gestaltungsregeln ergeben sich aus dem physikalisch-technischen Hintergrund einzelner Prozeßabschnitte und sind dadurch semantisch eng an diese gebunden. Aus der Sicht des Konstruierenden ist es daher sinnvoll, Gestaltungsregeln entsprechend ihres fertigungstechnischen Hintergrundes, also nach Verfahrensschritten zu klassifizieren [Leß92].

.

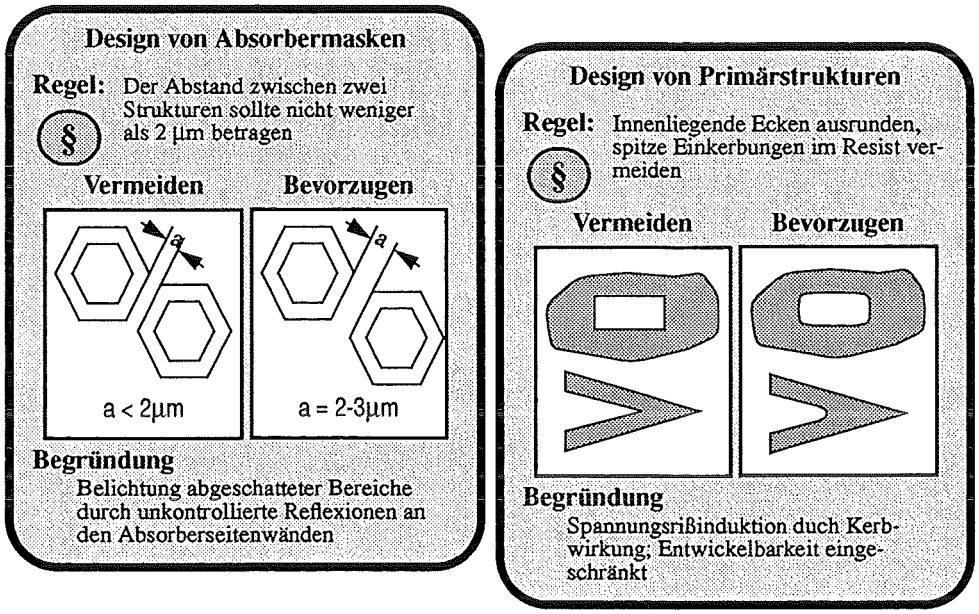


Abbildung 7: *Beispiele für Gestaltungsregeln*

Das in den frühen Konstruktionsphasen benötigte Fertigungswissen weist in der Regel eine semantisch völlig andersartige Struktur als die vorgestellten Gestaltungsregeln auf. So sind die in Tabelle 1 dargestellten Anhaltswerte für Produktabmessungen und Toleranzen beispielsweise nicht bestimmten Prozeßschritten zugeordnet, sondern hängen vom verwendeten Material ab. Im Gegensatz zu Gestaltungsregeln beinhalten sie rein numerische Informationen. Aus diesem Grund müssen die auf frühe Konstruktionsphasen abzielenden Gestaltungshinweise auf andere Weise formalisiert werden. Eine mögliche Formalisierung der in Abb.5 und Tabelle 1 dargestellten Informationen zeigt Abb.8.

Im Falle des in Abb.5 gezeigten Produkttypen wird die dazugehörige Produktgeome-

trie in Form eines Textes und einer Illustration dargestellt. Weitere Merkmale eines Produkttypen sind das verwendete Material, der dazugehörige Herstellungsprozeß und der Prozeßstand, der angibt, in welchem Entwicklungsstadium sich das Herstellungsverfahren befindet. Die Prozeßbeschreibung besteht aus einer Liste der zur Anwendung kommenden Prozeßschritte, die darin entsprechend ihrer Abfolge aufgeführt sind.

Die in Tabelle 1 erfaßten Anhaltswerte für Formate, Maße und Toleranzen beinhalten - neben einem Hinweis auf das Material - Grenzwerte für erreichbare Fertigungsmaße, wie das Aspektverhältnis oder die Höhe von Mikrostrukturen, und die bei der Fertigung einzukalkulierenden Toleranzen.

.

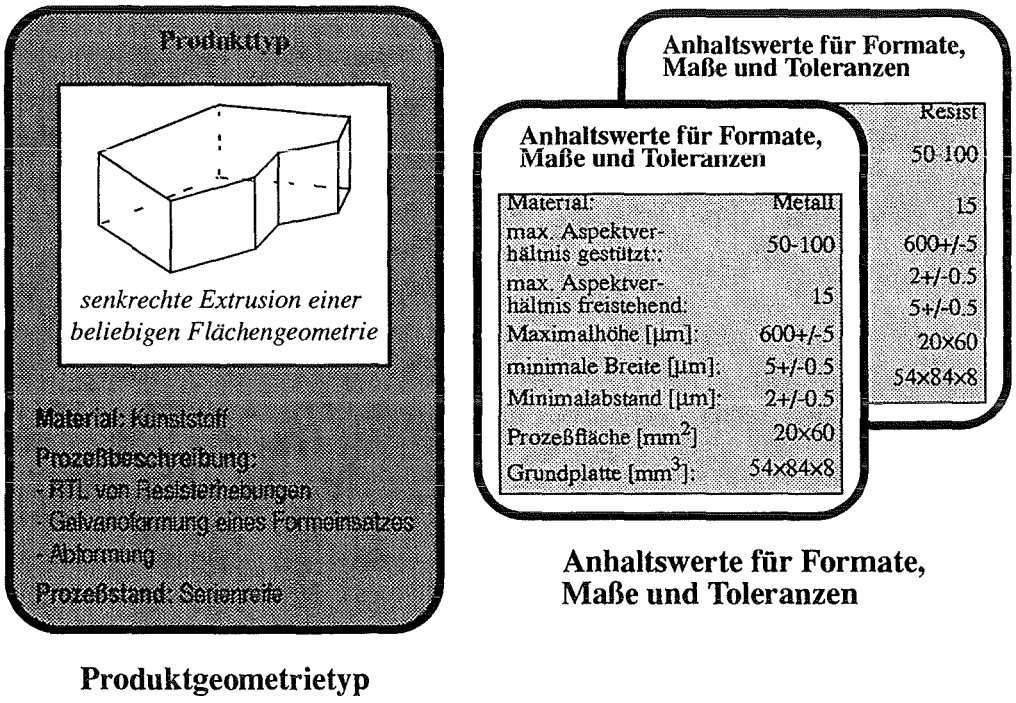


***Abbildung 8:** Beispiele für Gestaltungshinweise*

3.1.2 Algorithmische Klassifizierungskriterien

Während die im vorangegangenen Abschnitt vorgestellten semantischen Klassifizierungskriterien für die Schnittstelle zwischen Anwender und Maschine von Bedeutung sind, stehen im Rahmen einer rechnergestützten Anwendung von Regelwissen, z.B. in Form einer Entwurfsdiagnose, völlig andere Fragestellungen im Vordergrund.

Das Hauptproblem ist in diesem Zusammenhang die Umsetzung der in linguistischer

Form vorliegenden Gestaltungsregeln in entsprechende Algorithmen zur Entwurfsregelprüfung. Eine wesentliche Rolle spielt hierbei die sogenannte *algorithmische Prüfbarkeit* der anzuwendenden Regeln [Hahn95], also die Frage inwieweit und auf welche Weise Entwurfsregeln in eine algorithmische Form überführt werden können.

Hinsichtlich der zur ihrer Überprüfung auszuführenden Aktionen kann man die in [Leß92] erfaßten Entwurfsregeln in folgende Katetgorien untergliedern:

- **Formale CAD-bezogene Regeln**
  Die spezielle Geometrieverarbeitung der bei der Maskenerstellung eingesetzten Mustererzeugungsmaschinen (pattern generators) erfordert die Einhaltung einiger formaler geometrischer Regeln beim Entwurf. So darf eine Konstruktionszeichnung beispielsweise keine Flächenüberschneidungen enthalten, da ansonsten die Gefahr von Doppelbelichtungen besteht, bzw. selbstüberschneidende Polygone im Postprocessing entfernt werden. Ebenso sollten alle Figuren geschlossene Umrandungen aufweisen.
  Die Einhaltung dieser formalen Regeln ist eine notwendige Voraussetzung für die weitere Überprüfung eines Entwurfes und sollte unabhängig vom zur Anwendung kommenden Verfahren gewährleistet sein. Es ist daher erforderlich, der eigentlichen Entwurfsdiagnose die Überprüfung dieser formalen Kriterien in einem Preprocessing voranzustellen. Aus diesem Grund wurde am Forschungszentrum Karlsruhe für diesen Zweck im Rahmen des LIDES-Systems (Liga Design and Engineering System) ein eigenes unabhängiges Werkzeug entwickelt [Eg95]. Dieses wird dort bereits am Institut für Mikrostrukturtechnik für formale Überprüfung von Layouts eingesetzt.

- **Einfache geometrische Regeln**
  Einfache geometrische Regeln wie die in Abb.7 vorgestellte Abstandsregel beziehen sich auf elementare geometrische Merkmale quantitativen Charakters. Hierzu zählen unter anderem:
  - Höhe und Dicke einer Mikrostruktur
  - Grundfläche und Flächenmomente einer Struktur
  - Gesamtfläche aller Strukturen
  - Aspektverhältnis
  - Winkel
  - Höhe benachbarter Strukturen
  - Abstände zwischen Strukturen
  - Krümmungsradien

  Die Aussage einer Regel ist bei dieser Regelkategorie äquivalent zur Vorgabe eines Sollwertes oder einzuhaltenden Wertebereiches für das betroffene Merkmal, d.h. die Einhaltung einer Regel ist direkt durch Vergleich von Soll- und Ist-Werten verifizierbar.

- **Verteilungsregeln**
  Verteilungsregeln beziehen sich im Gegensatz zu den bereits vorgestellten Regeln nicht auf einzelne geometrische Merkmale einer Mikrostruktur, sondern betreffen die Anordnung der Strukturen auf dem Substrat.

  Beispiel 1: Fertigungsgerechte Gestaltung von Formeinsätzen

  Bei der Erstellung von Formeinsätzen ist es beispielsweise sinnvoll, biegeanfällige Strukturen, möglichst zum Zentrum hin auszurichten, um eine höhere Stabilität bei der Schwindung des Kunststoffes zu erreichen (s. Abb.9), die beim Abkühlen der Formmasse auftritt. Die Kontraktionen des Kunststoffmaterials bewirken eine Seitenbelastung der zum Formeinsatz gehörenden Strukturen und damit zu höheren Abformkräften.

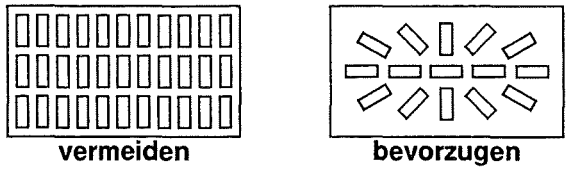


Abbildung 9: Beispiel für die fertigungsgerechte Gestaltung von Formeinsätzen

Beispiel 2: Entwurf von Röntgenmasken

Bei der Gestaltung von Röntgenmasken sollten wichtige Strukturen mit hohen Präzisionsanforderungen bevorzugt in die Randbereiche verlegt werden, da sich dort thermische Verzüge der Maskenmembran weniger stark auswirken und die dort plazierten Strukturen dementsprechend genauer abgebildet werden.

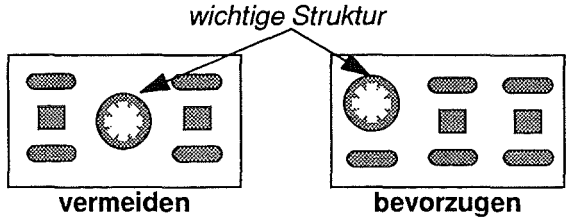


Abbildung 10: Beispiel für die fertigungsgerechte Gestaltung von Formeinsätzen

Die Umsetzung derartiger Verteilungsregeln in entsprechende Prüfalgorithmen ist im allgemeinen schwieriger und mit größerem Aufwand zu realisieren als dies z.B. bei der Betrachtung einfacher geometrischer Merkmale der Fall ist. Verteilungsregeln erfordern einerseits die Beurteilung der gesamten Substratfläche, andererseits sind die heranzuziehenden Beurteilungskriterien in ihrem semanti-

schen Gehalt oft nur schwer oder kaum in numerischer Form zu erfassen, wie bei der in Beispiel 2 vorgestellten Regel, die eine Unterscheidung zwischen „wichtigen" und „unwichtigen" Strukturen beinhaltet.

- **Regeln zu Hilfs- und Teststrukturen**
  In vielen Fällen ist es günstig, den Fertigungsablauf durch zusätzliche Konstruktionselemente wie Hilfs- oder Teststrukturen, die keine Verwendung im Rahmen des Endproduktes finden, zu unterstützen.
  Unter dem Aspekt eines entwicklungsgerechten Entwurfs von Mikrostrukturen sollten beispielsweise Sacklöcher im Resist vermieden werden. Man kann die Entwickelbarkeit verbessern, indem man Stofftransport und -austausch für das Entwicklermedium durch geeignete konstruktive Maßnahmen wie Verbindungsgräben unterstützt (siehe Abb.11).

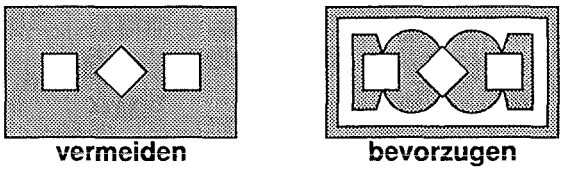


Abbildung 11: Beispiel für eine konstruktive Maßnahme zur Unterstützung der Entwicklung

Für die Überwachung des Fertigungsprozesses kann es außerdem sinnvoll sein, zusätzliche Teststrukturen an geeigneter Stelle auf dem Substrat zu plazieren.
Hinsichtlich der algorithmischen Prüfbarkeit gelten für diese Kategorie von Regeln im wesentlichen die bei Verteilungsregeln genannten Punkte. Da ein erheblicher Anteil dieser Regeln auf der Unterscheidung zwischen „Nutzstrukuren" und „Hilfsstrukturen" basiert, ist bei diesen eine algorithmische Überprüfung nur möglich, wenn das zur Darstellung einer Entwurfsgeometrie verwendete Datenmodell auch Elemente zur Beschreibung dieser semantischen Aspekte bereitstellt.

3.1.3 Konsistenz und Wertigkeit von Gestaltungsregeln

Wie die folgenden Beispiele aus den Arbeitsschritten Entwicklung und Vereinzelung zeigen, können sich aus den einzelnen Prozeßabschnitten sehr unterschiedliche oder sogar widersprüchliche Randbedingungen für die Gestaltung von Mikrostrukturen ergeben.

Beispiel 1: Entwicklung

Ein bei der Entwicklung belichteten Resistmaterials auftretendes Problem sind die bei der Aufpolymerisation des Resists eingefrorenen Eigenspannungen. Nach dem Entwickeln verteilen sich diese Spannungen auf die übrigen Resistteile und können dort zur Bildung von Rissen führen. Die Spannungsrißgefahr kann u.a. dadurch reduziert werden, daß man Kanten von Resistvertiefungen

nicht auf dieselbe Höhe legt (s. Abb.12).

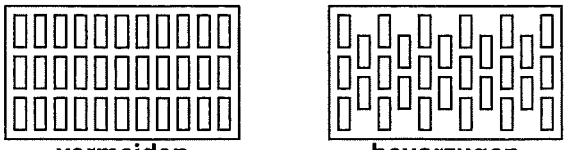


Abbildung 12: Beispiel für eine Maßnahme zur Vermeidung von Spannungsrissen

Beispiel 2: Vereinzelung
Bei der Vereinzelung durch Sägen oder Fräsen sind in der Regel nur geradlinige Schnitte möglich. Daher sollten die Mikrostrukturen im Raster angeordnet werden, d.h. die Kanten der Strukturen sollten auf derselben Höhe liegen (s. Abb.13).

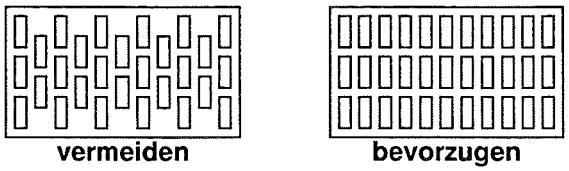


Abbildung 13: Beispiel für eine konstruktive Maßnahme zur Unterstützung der Vereinzelung

Während die in Beispiel 1 vorgestellte Regel zur Vermeidung von Spannungsrissen eine versetzte Anordnung der Strukturen empfiehlt, ist - wie Beispiel 2 zeigt - bei der Vereinzelung genau die gegenteilige Anordnung zu bevorzugen, d.h. die Strukturen sollten im Raster angeordnet werden. Für die Gestaltung von Formeinsätzen ist entsprechend der in Abb.10 dargestellten Gestaltungsregel wieder eine andere Art der Verteilung empfehlenswert, die aber mit den beiden anderen Regeln zumindest teilweise in Einklang gebracht werden kann, da sie sich nur auf die Ausrichtung von Strukturen bezüglich des Substratzentrums bezieht.

Wie diese Beispiele belegen, sind die bei der Gestaltung von Mikrostrukturen zu beachtenden Regeln untereinander nicht notwendigerweise konsistent. Welcher Regel im Zweifelsfall der Vorzug zu geben ist, hängt von den zugrundeliegenden prozeßtechnischen Einflußfaktoren und den funktionalen Anforderungen an die zu fertigende Struktur ab. Unter Umständen besitzt eine Regel eher empfehlenden Charakter und kann ohne gravierende Folgen außer acht gelassen werden, in anderen Fällen mag sich die Einhaltung einer Regel dagegen als unabdingbar erweisen. Daher ist es sinnvoll, Regeln für den Fall von Konfliktsituationen eine Prioritätsstufe

zuzuordnen, anhand derer eine Unterscheidung zwischen fertigungstechnisch wichtigen und weniger zwingenden Randbedingungen möglich ist.

3.2 Anforderungen an die Modellierung des Fertigungswissens

Die Anforderungen an die Modellierung des Fertigungswissens sind bei MIDAS von den beiden spezifischen Aufgabestellungen der grundlegenden Systemkomponenten geprägt - der Darstellung von Fertigungswissen im Rahmen eines Informationssystems und der Anwendung von Regelwissen in Form einer Entwurfsdiagnose.

Das Informationssystem hat die Aufgabe, Fertigungswissen in einer der Sichtweise des menschlichen Anwenders angemessenen Form zu repräsentieren. Das der Informationskomponente zugrundeliegende Modell des Fertigungswissens muß sich daher an semantischen Kriterien orientieren. Wie die entsprechende Analyse in Abschnitt 3.1.1 zeigt, umfaßt das während der Konstruktion benötigte Fertigungswissen mehrere semantisch verschiedenartige Bereiche. Während früher Konstruktionsphasen benötigt der Konstrukteur vorwiegend Informationen allgemeineren Charakters, die Aufschluß über fertigbare Produkte und die Möglichkeiten des Verfahrens geben. Beispiele hierfür sind Anhaltswerte für Fertigungsmaße und Toleranzen oder die Beschreibung von Produkttypen.

In der Entwurfsphase stehen für den Konstrukteur die zu berücksichtigenden fertigungstechnischen Einflüsse im Vordergrund. Eine übliche Form zur Übermittlung derartiger Randbedingungen sind Gestaltungsregeln, die aus semantischer Sicht entsprechend ihres fertigungstechnischen Hintergrunds, also nach Prozeßschritten klassifiziert werden können.

Das dem Informationssystem zugrundeliegende Modell muß dementsprechend zweierlei ermöglichen - zum einen sind bei der Modellierung semantisch verschiedenartige Informationstypen wie Gestaltungsregeln oder Produkttypen zu berücksichtigen, zum anderen muß das Modell auch die Zusammenhänge zwischen verschiedenen Objekten wie z.B. Prozeßschritten und den von ihnen abgeleiteten Gestaltungsregeln wiedergeben können.

Bei der Diagnose steht die Anwendbarkeit der Gestaltungsregeln im Rahmen einer automatischen Entwurfsregelprüfung im Vordergrund. Aus den in linguistischer Form vorliegenden Regeln müssen dazu geeignete, algorithmisch prüfbare Größen extrahiert werden. Im Falle einfacher geometrischer Regeln, entspricht dies der Vorgabe eines numerischen Wertebereiches, die Einhaltung formaler Kriterien kann in Form von Boole'schen oder linguistischen Werten ausgedrückt werden.

Aufgrund unterschiedlicher zu berücksichtigender Einflußfaktoren kann es zu Konsistenzkonflikten innerhalb der Regelmenge kommen. Aus diesem Grund sollte die Modellierung von Entwurfsrandbedingungen zur Konfliktvermeidung auch die Vergabe von Prioritäten gestatten.

4. Die Informationskomponente von MIDAS

4.1 Problemstellung bei der Modellierung des Fertigungswissens

Wie die Erfahrungen auf dem Gebiet wissensbasierter Systeme zeigen [FK85], hängt die Effektivität derartiger Programme in starkem Maße von einer geeigneten Repräsentation der zugrundeliegenden Wissensdomäne ab. Jede Wissensdomäne besitzt ihre eigene Terminologie und semantische Struktur. So ist beispielsweise die Denkweise eines LIGA-Konstrukteurs im wesentlichen vom Ablauf des LIGA-Verfahrens, der Abfolge von Prozeßschritten, verschiedenen Prozeßalternativen und anderen Begriffen aus dem Prozeßumfeld geprägt.

Da die Funktionalität eines Informationssystems in starkem Maße von den darzustellenden Informationen beeinflußt wird, ist es wichtig, daß derartige semantische Strukturen in adäquater Weise bei der Modellierung des Wissens berücksichtigt werden. Wesentliche Kriterien hierfür sind:

- **die Ausdruckskraft der Wissensdarstellung**
  Die gewählte Repräsentationsform muß einen effektiven Informationsaustausch zwischen System und Experten, bzw. Anwender ermöglichen.
- **die Verständlichkeit der Darstellung**
  Das im System erfaßte Wissen muß in einer dem Anwender verständlichen Form wiedergegeben werden.
- **die Umsetzbarkeit des erfaßten Wissens**
  Das System muß in der Lage sein, die ihm gegebenen Informationen zu verarbeiten, d.h. vom menschlichen Anwender kommende Informationen müssen in eine maschinell auswertbare Form überführbar sein.

4.2 Zur Wissensdarstellung verwendete Methode

Wissen besteht nicht ausschließlich aus Daten, sondern beinhaltet immer auch strukturelle Informationen und prozedurale Elemente. Im Bereich wissensbasierter Systeme haben sich verschiedene Repräsentationsformen von Wissen etabliert, bei denen diese einzelnen Teilaspekte je nach Anwendungszweck unterschiedlich gewichtet werden.

Im Falle des Informationssystems stehen in erster Linie die Auswahl und Visualisierung von Informationen im Vordergrund. Wie die Analyse des Fertigungswissens gezeigt hat, beinhaltet dieses zwar auch verschiedene Formen von Regeln, jedoch ist die in Abb.8 gezeigte textuelle, bzw. graphische Beschreibungsform für den Informationsaustausch mit dem Anwender geeigneter als eine regelbasierte, z.B. auf Prädikatenlogik [ThSc89] beruhende Darstellungsweise.

Die Erfahrungen im Bereich der Wissensbearbeitung haben gezeigt, daß die menschliche Sichtweise eines Problemfeldes von den darin enthaltenen Objekten

und den zwischen diesen Objekten bestehenden Beziehungen geprägt ist [CY91]. Daher ist es im Rahmen des Informationssystems sinnvoll, bei der Modellierung des Fertigungswissens auf eine entsprechende objektorientierte Beschreibungsform zurückzugreifen.

4.3 Objektorientierte Modellierung des Fertigungswissens

Ziel eines objektorientierten Ansatzes ist es, die zu modellierenden Aspekte der realen Welt auf eine Menge von Objekten mit definierten Attributen und Aufgaben abzubilden. Die Anzahl und Art der hierbei benötigten Objektattribute hängt jeweils von der semantischen Struktur des zu repräsentierenden Wissensbereiches ab. Semantisch gleichartige Objekte werden zu Klassen zusammengefaßt.

Mit *Gestaltungsregeln*, *Produkttypen*, sowie *Anhaltswerten für Maße oder Toleranzen* wurden im vorangegangenen Kapitel einige wesentliche Formen des zu modellierenden Fertigungswissens vorgestellt. Die dabei erarbeitete formalisierte Darstellung dieser Objekte kann direkt in ein Klassenschema mit Attributen eines geeignet gewählten Datentyps übertragen werden (siehe Abb.14). Entsprechend seinem realen Vorbild besitzt beispielsweise ein Objekt der Klasse „Gestaltungsregel" in diesem Modell einen *Namen*, eine *Regelformulierung* und eine *Begründung* als textuelle Attribute, sowie die Abbildungen *günstiger*, bzw. *ungünstiger Entwurfskonstellationen* als graphische Attribute.

Die wichtigste Aufgabe dieser Objekte besteht im Rahmen eines Informationssystems darin, Wissensinhalte mit Hilfe einer Benutzeroberfläche in geeigneter Form darzustellen. Für die Wissensakquisition benötigt man außerdem eine entsprechende Eingabemöglichkeit. Unter dem Aspekt einer persistenten Speicherung von Wissensinhalten müssen die Objekte zusätzlich über eine Ein-, bzw. Ausgabeschnittstelle zur Datenbasis verfügen. Die dafür benötigten Prozeduren stellen die Klassen selbst als entsprechende Methoden zu Verfügung.

Eine objektorientierte Modellierung beinhaltet jedoch nicht nur die Bildung von Klassen, sondern umfaßt auch die Beziehungen zwischen Objekten, bzw. Klassen. Bei den wichtigsten dieser Relationen handelt es sich um „ist ein" („is a" oder „is a kind of") und „ist Teil von" („is part of").

Mit Hilfe der „is a"-Relation können hierarchische Strukturen aus Klassen aufgebaut werden, bei denen eine abstrakte Klasse immer mehr spezialisiert wird. Die Eigenschaften der übergeordneten Klasse werden dabei an die von ihr abgeleiteten Unterklassen vererbt. In Form von „is part of"-Relationen werden die Bestandteile eines Objektes spezifiziert.

Wie die Analyse des für den Entwurf von LIGA-Strukturen benötigten Fertigungswissens zeigt, hat man es hierbei mit semantisch verschiedenartigen Wissensbereichen und dementsprechend auch mit einer heterogen ausgeprägten Klassenstruktur zu tun. Die einzige Gemeinsamkeit zwischen Objekten wie Gestaltungsregeln oder Pro-

dukttypen besteht darin, daß es sich um spezielle Formen von Fertigungswissen handelt, die über Eingabe-, bzw. Ausgabschnittstellen zu Benutzeroberfläche und Datenbasis verfügen.

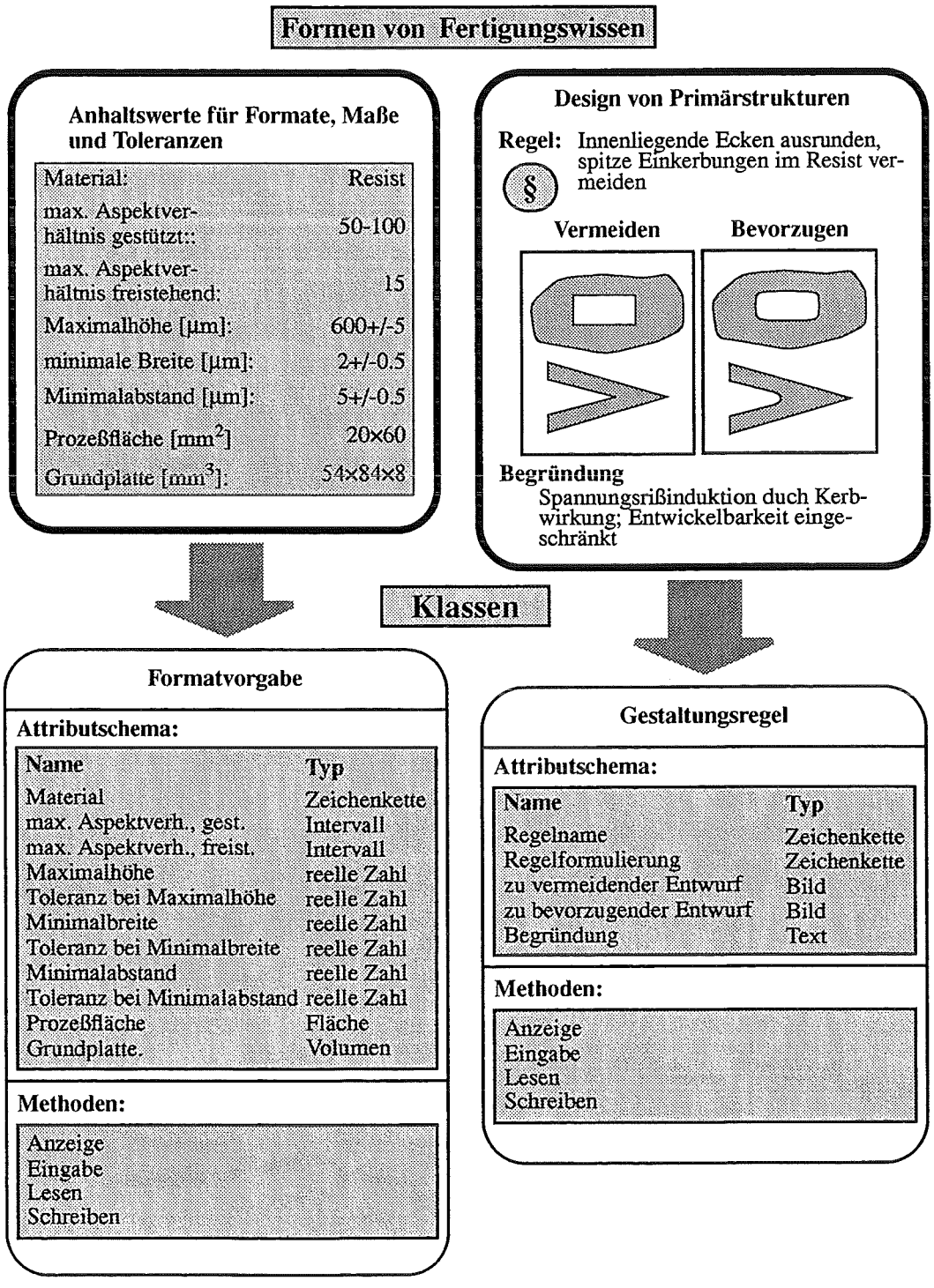


Abbildung 14: *Bildung von Klassen*

Bei der objektorientierten Modellierung setzt man dies in eine entsprechende Klassenhierarchie um, an deren Spitze sich die abstrakte Basisklasse „Fertigungswissen" befindet. Von dieser Klasse werden alle spezielleren Formen von Fertigungswissen wie die Klassen „Regel" oder „Produkttyp" abgeleitet.

Abb.15 gibt die vorgestellte Klassenstruktur in der bei Coad/Yourdon [CY92] üblichen Notation wieder.

Im Falle der Klasse „Regel" sind noch weitere Verfeinerungen möglich. Das Modell unterscheidet hierbei zunächst zwischen Regeln zur Materialauswahl und Gestaltungsregeln. Letztere können wiederum entsprechend ihres fertigungstechnischen Bezugsrahmens, also nach ihrer Zugehörigkeit zu bestimmten Prozeßschritten klassifiziert werden. Demgemäß beinhaltet das Modell Regeln für den Entwurf von Röntgenmasken, Primärstrukturen, etc.

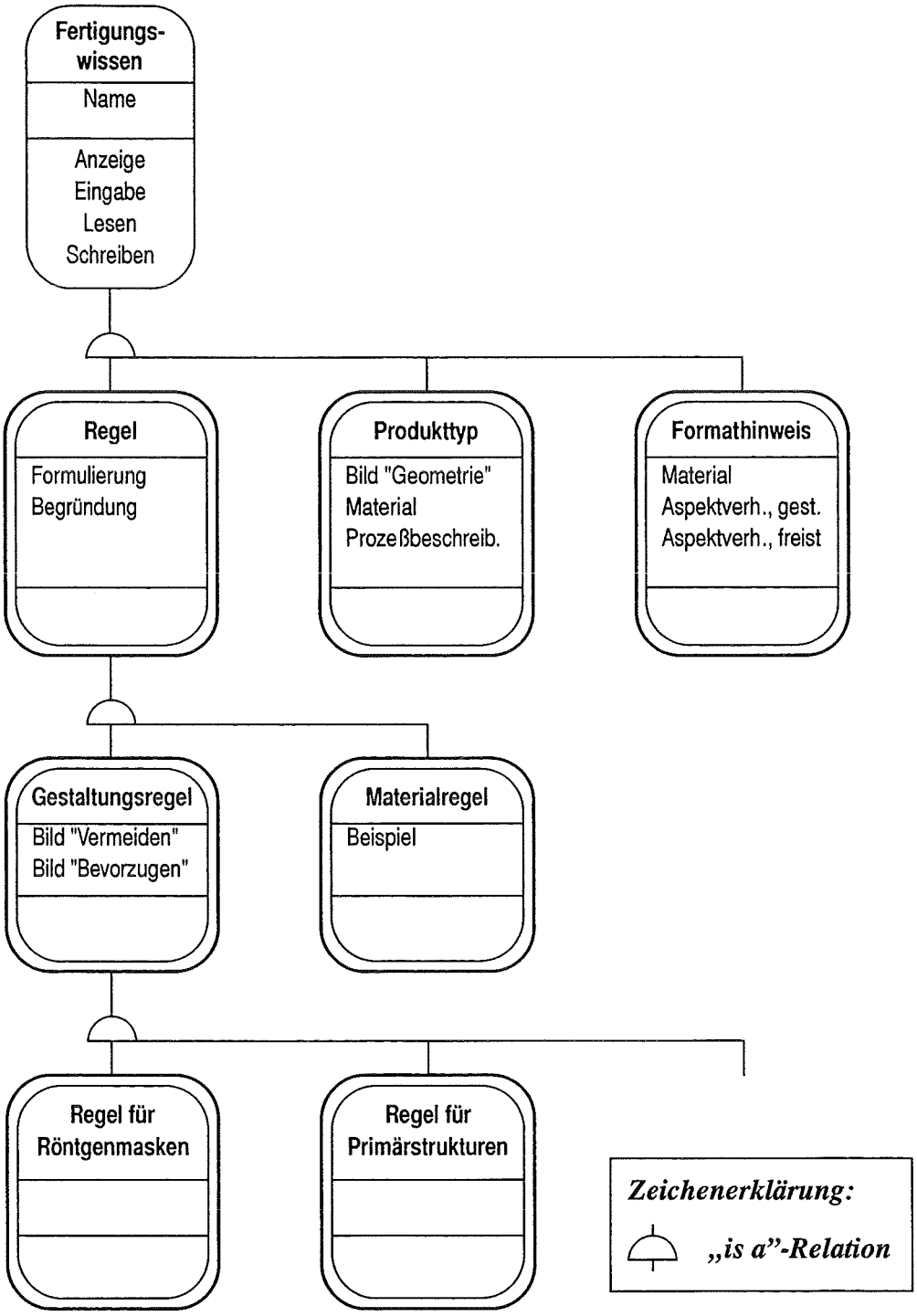


Abbildung 15: *Klassenstruktur des Fertigungswissens*

4.4 Aufbau des Informationssystems

Das Informationssystem soll dem Benutzer einen gezielten Zugriff auf die in der Wissensbasis enthaltenen Klassen und ihre Instanzen ermöglichen. Hierzu wird ein Navigationswerkzeug benötigt, das dem Anwender erlaubt, sich innerhalb einer Wissensbasis zu bewegen.

Der hierfür entwickelte Navigationsmechanismus basiert auf den drei in Abb.16 dargestellten Zugriffsebenen.

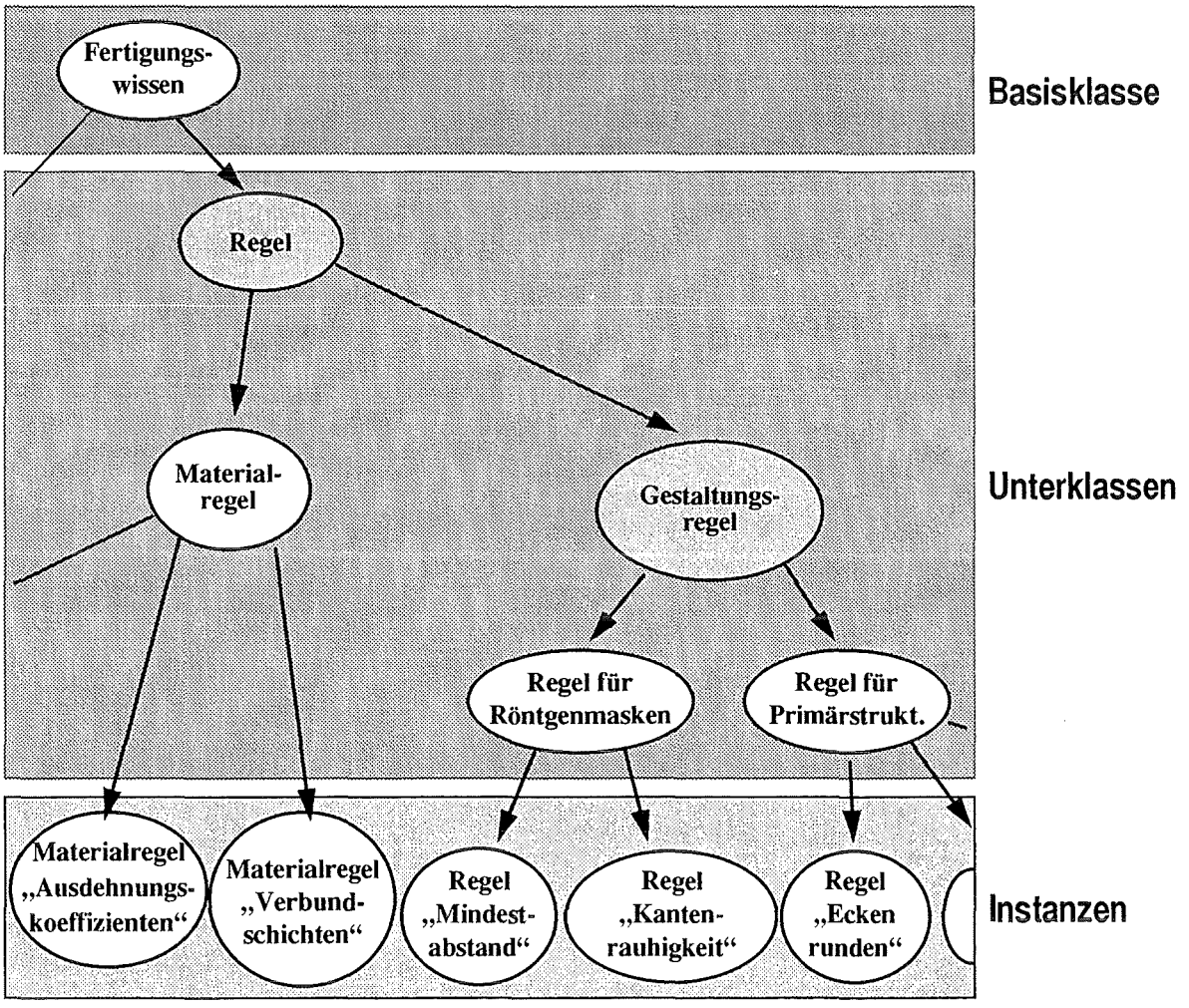


Abbildung 16: *Zugriffsebenen des Informationssystems*

Auf der obersten, abstraktesten Ebene befindet sich die Basisklasse als Ausgangspunkt der hierarchischen Struktur. Diese verzweigt sich zur zweiten Ebene des Modells hin in verschiedene Klassen von Fertigungswissen. Auf der untersten Ebene des Modells befinden sich die zu diesen Klassen gehörigen Instanzen.

Entsprechend dieser Schichtstruktur besteht der Navigationsmechanismus aus zwei Kontrollanzeigen, die es dem Benutzer erlauben, sich vertikal in der Baumstruktur der Wissensbasis zu bewegen. Jede Kontrollanzeige enthält eine Übersicht über die auf der nächsttieferen Ebene erreichbaren Knotenpunkte. Am Ausgangspunkt, also auf der Ebene der Basisklasse, erhält man dementsprechend eine Übersicht über die vorhandenen Unterklassen, wobei nur instanziierbare Klassen berücksichtigt werden. Durch Auswahl einer Klasse gelangt man in deren Ebene und erhält eine Übersicht über ihre Instanzen. Bei Selektion einer Instanz wird deren Visualisierungsmethode aufgerufen, d.h. ihre Attributwerte werden angezeigt.

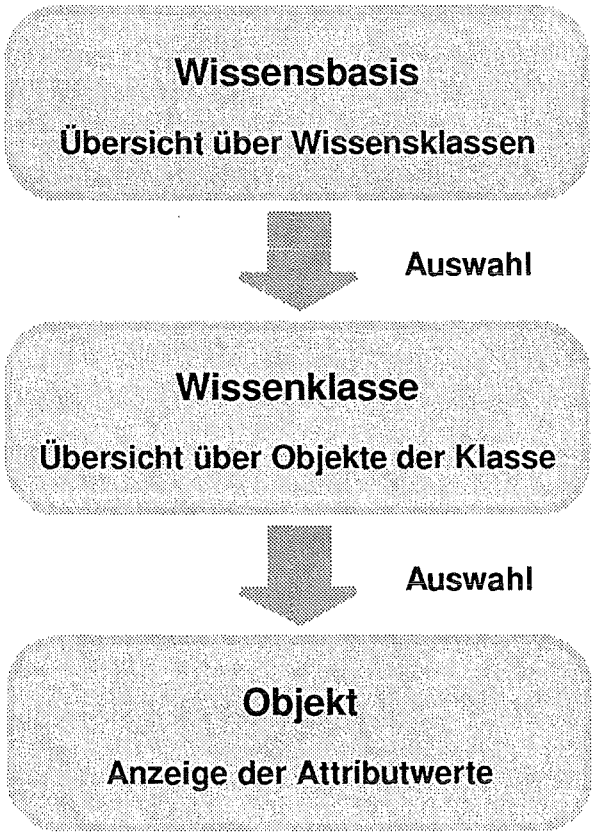


Abbildung 17: *Navigationsmechanismus des Informationssystems*

Bewegungen innerhalb einer Ebene sind jeweils nur über höhere Verzweigungspunkte möglich, d.h. um von einer Klasse in die nächste zu gelangen, muß der Benutzer eine entsprechende Auswahl auf der Ebene der Basisklasse vornehmen.

4.5 Werkzeug zur Wissensakquisition

Bedingt durch die fortschreitenden Entwicklung des LIGA-Verfahrens unterliegt das hierzu verfügbare Fertigungswissen häufigen Änderungen, die eine entsprechende Aktualisierung der Wissensbasis erfordern.

Diese Änderungen beziehen sich im einfachsten Fall nur auf die Inhalte einzelner Objekte, gegebenenfalls ist aber auch eine Modifikation der Klassenstruktur erforderlich. Aufgrund des engen prozeßtechnischen Bezugs der verschiedenen Entwurfsregelklassen erfordert zum Beispiel das Hinzufügen eines neuen Prozeß-schrittes unter Umständen auch die Einführung einer entsprechenden Entwurfsregelklasse.

Zur Wissensakquisition, bzw. zur Pflege der Wissensbasis wird daher ein Werkzeug benötigt, das einem autorisierten Benutzer sowohl auf Objekt- als auch Klassenebene Zugriff auf die Inhalte der Wissensbasis ermöglicht. Um eine möglichst schnelle und korrekte Erfassung des Fertigungswissens zu gewährleisten, sollte die Akquisition direkt von Prozeßexperten vorgenommen werden. Dies setzt eine an dessen Denkweise orientierte Gestaltung der Benutzeroberfläche voraus.

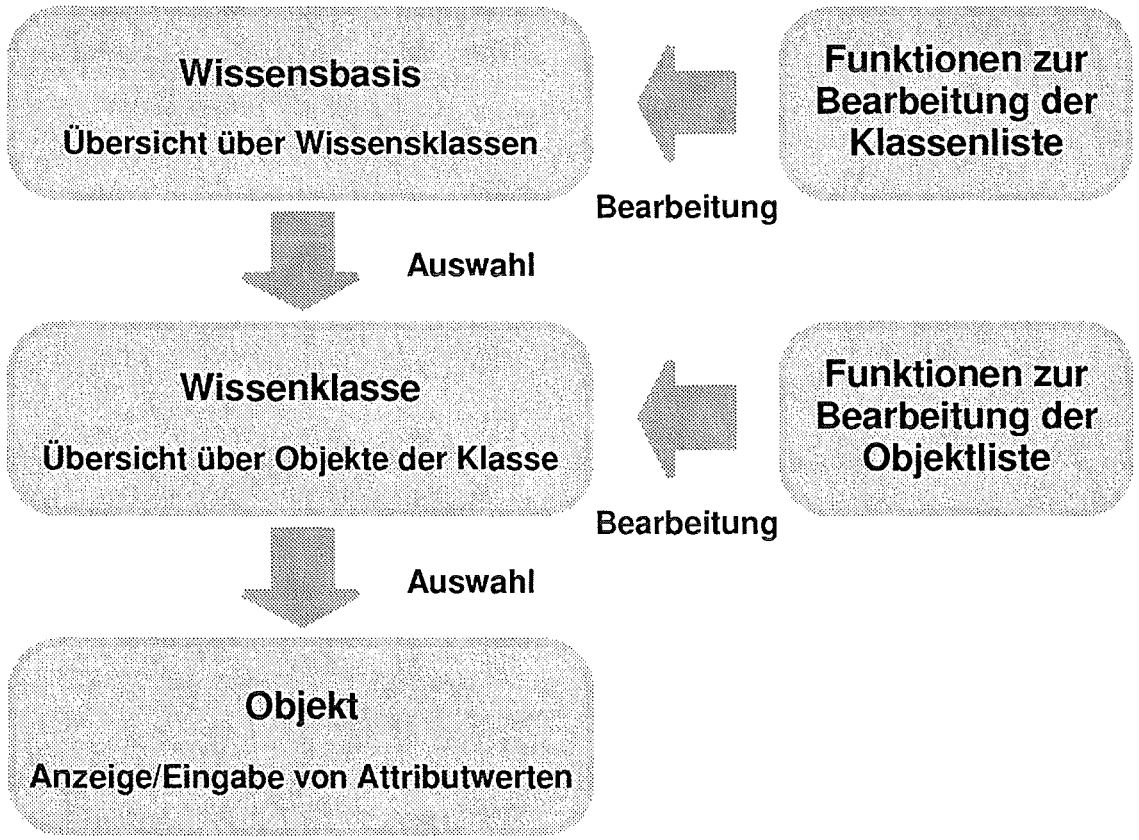


Abbildung 18: Aufbau des Akquisitionswerkzeuges

In seiner Aufgabenstellung ist dieses Werkzeug daher nahe mit dem Informationssy-

stem verwandt. In beiden Fällen muß der Anwender die Möglichkeit erhalten, gezielt in der Wissensbasis zu navigieren und Wissen abzufragen. Der wesentliche Unterschied zwischen beiden Werkzeugen besteht darin, daß der Benutzer des Akquisitionswerkzeuges in die Lage versetzt werden muß, auch Änderungen an den dargestellten Daten und Strukturinformationen vornehmen zu können.

Aufgrund der engen Verwandtschaft zwischen beiden Werkzeugen liegt der Wissenserwerbskomponente im wesentlichen dieselbe Architektur zugrunde wie dem Informationssystem (siehe Abb.18).

Der vom Informationssystem übernommene Navigationsmechanismus wird beim Akquisitionswerkzeug um zusätzliche Kontrollelemente zur Manipulation von Klassen- und Objektauswahl erweitert. Der Anwender kann damit bei Bedarf neue Listeneinträge einfügen oder bereits vorhandene modifizieren, bzw. löschen.

Zur Bearbeitung von einzelnen Attributwerten wird die zur Objektdarstellung verwendete Bildschirmanzeige für Benutzereingaben freigegeben. Der dabei zur Anwendung kommende Eingabemechanismus basiert auf den dafür von den Objekten bereitgestellten Eingabemethoden.

5. Die Diagnosekomponente von MIDAS

Bei einer Entwurfsdiagnose wird die Gestaltung einer Mikrostruktur unter dem Aspekt ihrer fertigungstechnischen Eignung beurteilt. Als Grundlage der Beurteilung dient ein in Anlehnung an das vorgesehene Fertigungsverfahren zusammengestellter Satz von Entwurfsregeln. Die Aufgabe der Diagnosekomponente von MIDAS besteht darin, die Einhaltung derartiger Randbedingungen bei einem Entwurf zu verifizieren und Designfehler zu protokollieren.

Ausgehend vom Ablauf einer Entwurfsdiagnose wird in den sich anschließenden Abschnitten die grundlegende Struktur der Diagnosekomponente beschrieben und entsprechend funktionaler Aspekte verfeinert. Im folgenden sollen zunächst die Basiskomponenten eines derartigen wissensbasierten Systems vorgestellt werden.

5.1 Die Architektur des Diagnosesystems

5.1.1 Basiskomponenten eines wissensbasierten Systems

Wissensbasierte Diagnosesysteme kommen in einer Vielfalt von Anwendungsgebieten zum Einsatz. Das Spektrum reicht dabei von den verschiedensten technischen [HK85] bis hin zu medizinischen Anwendungen [BuSh84].

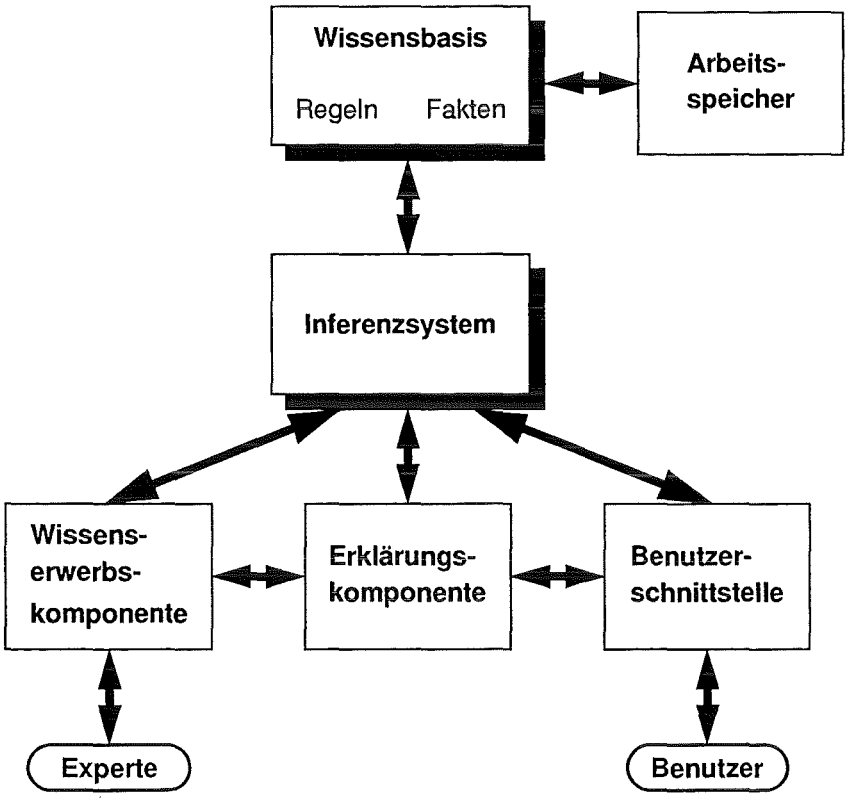


Abbildung 19: Struktur eines wissensbasierten Systems [HK86]

Trotz ihrer unterschiedlichen Einsatzgebiete finden sich verschiedene Grundelemente - wenn auch in unterschiedlicher Ausprägung - in allen Systemen wieder. Den typischen Aufbau eines wissensbasierten Systems zeigt Abb.19 [HK86,KKP87].

Den Kern des Systems bilden die Wissensbasis und die Inferenzkomponente. Die Wissensbasis enthält domänenspezifisches Regel- und Faktenwissen. Als Inferenzkomponente oder Inferenzmaschine bezeichnet man in der Begriffswelt wissensbasierter Systeme die für die Verarbeitung des Wissens zuständige Systemkomponente. Aufgabe einer Inferenzmaschine ist die Gewinnung neuen Wissens aus den bereits in der Wissensbasis erfaßten Inhalten und den gegebenen Falldaten, wobei je nach Einsatzbereich unterschiedliche Inferenzstrategien zur Anwendung kommen können.

Wissensbasierte Systeme verfügen in der Regel über zwei Schnittstellen zum menschlichen Anwender. Mit Hilfe der Wissenserwerbskomponente hat ein Experte die Möglichkeit, die Inhalte der Wissensbasis festzulegen oder zu modifizieren. Die Benutzerschnittstelle dient dem Dialog mit dem normalen Anwender des Systems. Die mit diesen Schnittstellen verbundene Erklärungskomponente hat die Aufgabe, Benutzeranfragen bezüglich der ausgeführten Informationsverarbeitung zu bearbeiten.

5.1.2 Ablauf einer Entwurfsdiagnose

Auf Basis der im vorigen Abschnitt vorgestellten Grundelemente soll im folgenden Schritt für Schritt die Architektur des Systems beschrieben werden. Ansatzpunkt ist hierbei der Ablauf einer Entwurfsdiagnose.

Für eine derartige Diagnose werden zunächst zweierlei Formen von Wissen benötigt - zum einen das anzuwendende Regelwissen, zum anderen das zu überprüfende Fallwissen, also die Werte relevanter Entwurfsparameter.

Die Erfassung von Entwurfsregeln erfolgt bei MIDAS mit Hilfe des im Rahmen des Informationssystems bereitgestellten Akquisitionswerkzeuges. Für die Diagnose muß aus der vorhandenen Regelmenge eine geeignete Untermenge selektiert werden. Welche Regeln zur Anwendung kommen, hängt vom jeweiligen Produkt ab.

Das zu überprüfende Fallwissen liegt bei einer geometrischen Überprüfung der Produktgestalt zunächst in Form einer im IGES-Format gegebenen CAD-Entwurfsdatei vor, die zur weiteren Bearbeitung in ein objektorientiertes Datenformat konvertiert wird. Daraus müssen die Werte der zu prüfenden Gestaltparameter extrahiert werden.

Im nächsten Schritt, der eigentlichen Diagnose, werden die ermittelten Parameterwerte mit den fertigungstechnisch sinnvollen Wertebereichen verglichen und eventuelle Verletzungen dieser Grenzwerte protokolliert.

Den Abschluß der Diagnose bildet die Interpretation des Fehlerprotokolles im Dialog mit dem Konstrukteur.

5.1.3 Der Aufbau des Diagnosesystems

Entsprechend des vorgestellten Ablaufs muß das Diagnosesystem folgende Funktionen beinhalten:

- Auswahl der zu beachtenden Entwurfsregeln (Regelselektion) aus der gegebenen Regelmenge
- Extraktion relevanter Geometrieparameter aus der Entwurfszeichnung (Merkmalsgenerierung)
- Vergleich von Soll- und Ist-Werten und Erstellen eines Fehlerprotokolles (Inferenz)
- Visualisierung des Fehlerprotokolles in Verbindung mit Verbesserungsvorschlägen (Fehlervisualisierung)

Das Diagnosesystem ist diesen Teilaufgaben entsprechend in unabhängige Module untergliedert, die über wohldefinierte Austauschschnittstellen miteinander kooperieren. Die daraus abgeleitete Architektur ist in Abb.20 wiedergegeben. Der Vorteil dieses modularen Systemaufbaus liegt zum einen in der Möglichkeit einer getrennten Entwicklung und Erprobung der Teilsysteme, zum anderen in der Austauschbarkeit der Module, die eine Anpassung des Systems an neue Anforderungen ermöglicht.

Das Herzstück des Systems bildet die Inferenzkomponente, deren Aufgabe in der Diagnose von Entwurfsdaten besteht.

Für die Bereitstellung des von der Inferenzmaschine benötigten Fakten und Regelwissens sind die Module zur Merkmalsgenerierung und zur Regelselektion zuständig. Während die Auswahl eines Regelsatzes im Dialog mit dem Anwender stattfindet, wird das benötigte Faktenwissen automatisch aus den Geometrieprimitiven des zu beurteilenden CAD-Entwurfes extrahiert und liegt danach in Form geometrischer Merkmale vor.

Die Interpretation der von der Inferenzkomponente erhaltenen Diagnoseresultate erfolgt im Dialog mit dem Benutzer. Das dafür vorgesehene Modul zur Fehlervisualisierung fungiert dabei als Erklärungskomponente. Es hat die Aufgabe, die ermittelten Konstruktionsfehler aufzuzeigen und dem Konstrukteur auf Anfrage eine fertigungstechnisch motivierte Begründung, sowie Vorschläge zu einer Verbesserung des Entwurfes zu geben.

Zur Steuerung des kompletten Diagnoseablaufs wird außerdem eine diesen Modulen übergeordnete Steuereinheit benötigt, über die der Benutzer Zugriff auf die einzelnen Systemkomponenten erhält.

Mit Aufbau und Funktionalität der einzelnen Systemkomponenten befassen sich die folgenden Abschnitte.

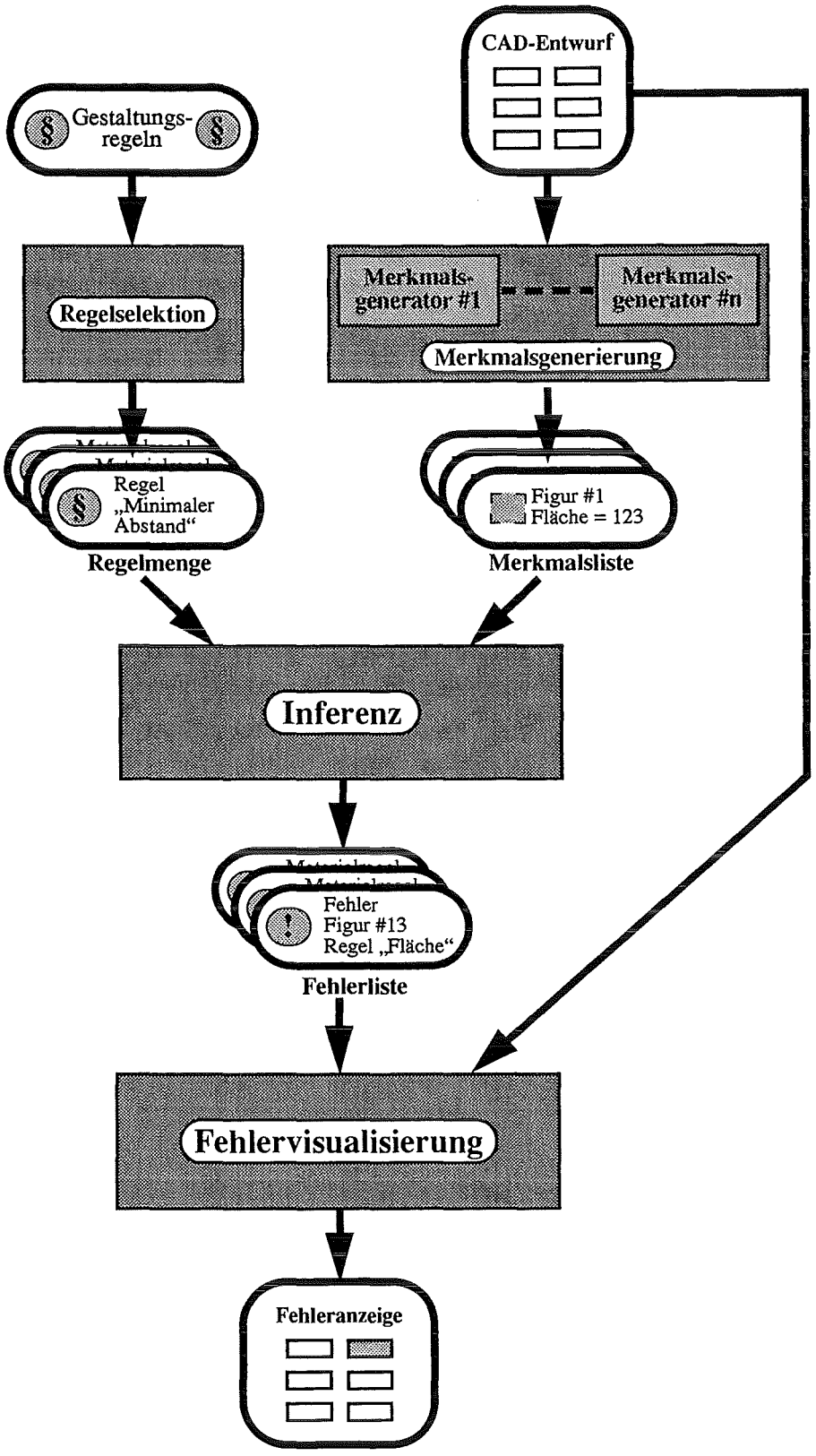


Abbildung 20: *Architektur des Diagnosesystems*

5.2 Vorgabe von Regelwissen

Entwurfsregeln werden bei MIDAS wie alle dem Fertigungswissen zugehörige Objekte mithilfe der für das Informationssystem entwickelten Klassen von Informationsträgern repräsentiert. Deren Verwendung im Rahmen einer Entwurfsdiagnose erfordert eine Erweiterung des dort vorgestellten Konzeptes um die Möglichkeit, eine beliebige Zahl von *algorithmisch prüfbaren* Entwurfsrandbedingungen mit einer Regel zu verknüpfen.
Unter diesem Gesichtspunkt ist es sinnvoll, bei der Modellierung auch auf regelbasierte Ansätze zurückzugreifen. Der folgende Abschnitt ist daher grundlegenden Elementen regelbasierter Konzepten gewidmet.

5.2.1 Grundelemente einer regelbasierten Wissensdarstellung

Eine vor allem in der Anfangszeit wissensbasierter Systeme häufig eingesetzte Technik zur Wissensdarstellung ist die sogenannte Prädikatenlogik. Diese Beschreibungsform beinhaltet sowohl daten- als auch prozedurartige Elemente [ThSc89].
Bei *Fakten* handelt es sich um von Bedingungen unabhängige Aussagen, d.h. sie besitzen datenartigen Charakter.
Prozedurales Wissen wird bei der Prädikatenlogik in Form von *Regeln* dargestellt. Regeln setzen sich aus einem Bedingungsteil, der die *Regelprämisse* beinhaltet, und einem *Konklusionsteil* zusammen (s. Abb.21). Die Regelprämisse kann dabei aus einer oder mehreren, durch logische Operatoren verknüpften, atomaren Bedingungen bestehen. Der Konklusionsteil enthält die bei Erfüllung aller Bedingungen zutreffende Schlußfolgerung, bzw. mehrere durch die „und"-Operation verknüpfte Schlußfolgerungen.

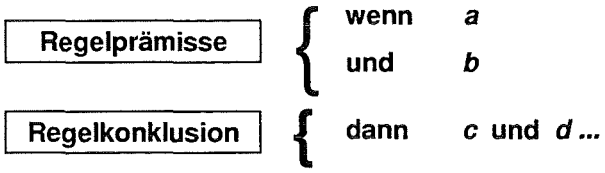


Abbildung 21: Schema einer Regel

5.2.2 Darstellung von Entwurfsregeln in Form der Prädikatenlogik

Auf die vorgestellten Typen von Entwurfsregeln läßt sich dieses Schema in folgender Weise übertragen, das in Abb.22 dargestellte Beispiel zeigt die Umsetzung der in Abb.7 vorgestellten Abstandsregel:

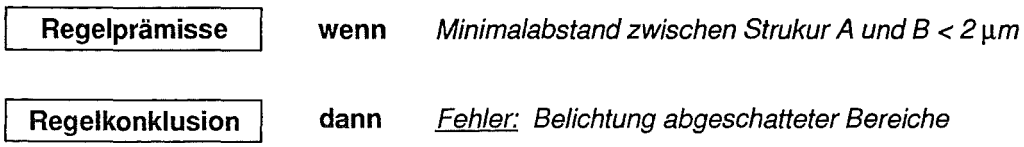


Abbildung 22: Darstellung einer Abstandsregel

Die Regelprämisse beinhaltet Überprüfung von Randbedingungen für einen oder mehrere geometrische Parameter. Im Falle numerischer Parameter wie Abständen, Winkeln oder Flächen besteht die Prämisse aus der Vorgabe eines geeigneten Sollwertbereiches für diese Parameter. Dabei ist Faktenwissen in Form des konkreten Parameterwertes zu überprüfen. Bei der in Abb.22 dargestellten Abstandsregel darf beispielsweise ein der Mindestabstand von 2 µm nicht unterschritten werden. Im Falle formaler Randbedingungen, wie z.B. bei der geforderten Geschlossenheit einer Figur oder der Verteilung von Mikrostrukturen auf dem Substrat, ist entsprechend ein Boole'scher Wert als Sollwert vorzugeben.

Bei Verletzung dieser Randbedingungen gelten die im Konklusionsteil enthaltenen Aussagen. Im vorliegenden Beispiel bedeutet dies, daß die betroffenen Strukturen A und B aufgrund unkontrollierter Reflexionen von Röntgenstrahlen, die zur Belichtung abgeschatteter Substratbereiche führt, Probleme bei der Fertigung bereiten. Das Diagnosesystem muß an dieser Stelle eine Fehlermeldung an den Konstrukteur ausgeben.

5.2.3 Modellierung von Entwurfsrandbedingungen

Bei einer für die Diagnose geeigneten Repräsentation von Entwurfsrandbedingungen müssen sich die genannten Grundelemente im Modell wiederfinden. Die Regelprämisse beinhaltet immer zwei Komponenten - aus dem Entwurf abgeleitetes Faktenwissen, also die Werte bestimmter Entwurfsparameter, und eine dieses spezielle Faktenwissen betreffende Randbedingung in Form eines geeignet vorgegebenen Wertebereichs.

Während das Faktenwissen bei jedem Entwurf neu generiert werden muß und semantisch eine eigene Form von Wissen darstellt, handelt es sich bei den Randbedingungen aus semantischer Sicht um den Bestandteil einer Regel. Aus diesem Grund wurde die im Rahmen des Informationssystems entwickelte Regelklasse um ein zusätzliches Attribut, einen Satz von Entwurfsrandbedingungen, erweitert.

Für eine vollständige Beschreibung einer Randbedingung müssen folgende Informationenvorliegen:

- **Parametertyp**
  Randbedingungen beziehen sich auf einen bestimmten Typ von Faktenwissen, also spezielle Arten von Entwurfsparametern, wie z.B. Abstände, Winkel oder Flächen. Das Modell muß dementsprechend die Beschreibung des Parametertyps in Form eines geeigneten Identifikators ermöglichen. Da sowohl die Erfassung als auch die Auswertung der Diagnoseresultate im Dialog mit dem Anwender eine Visualisierung von Randbedingungen erfordern, ist in diesem Zusammenhang die Verwendung einer textuellen Beschreibung, z.B. in Form von Schlüsselbegriffen wie „Abstand", „Winkel", etc. sinnvoll.
  Für die maschinelle Auswertung von Randbedingungen ist dem wegen des geringeren Rechenzeitaufwandes die Verwendung von numerischen Identifikato-

ren vorzuziehen. Beide Beschreibungsformen können jedoch z.B. mit Hilfe einer entsprechenden Übersetzungstabelle parallel verwendet werden.

- **Datentyp**
  Wie die Analyse des Fertigungswissens gezeigt hat, beziehen sich die beim Entwurf zu berücksichtigenden Randbedingungen nicht nur auf reellwertige Parameter; in eine Entwurfsdiagnose sind unter anderem auch ganzzahlige oder Parameter Boole'schen Typs einzubeziehen. Daher benötigt man zur Beschreibung von Randbedingungen einen Identifikator für den Datentyp.
- **Sollwertebereich**
  Für die Diagnose selbst ist der einzuhaltende Sollwertebereich wichtig. In welcher Form der Wertebereich repräsentiert wird, hängt vom gegebenen Datentyp ab. Hierbei kann es sich beispielsweise um ein ganzzahliges oder reellwertiges Intervall, gegebenenfalls verbunden mit einer Maßeinheit handeln.
- **Prioritätsstufe**
  Viele der in [Leß92] vorgestellten Entwurfsregeln besitzen eher empfehlenden Charakter, in manchen Fällen ist die Einhaltung einer Regel aber auch fertigungstechnisch notwendig. Aus diesem Grund ist es sinnvoll, die Bedeutung einer Regel, bzw. Randbedingung in Form einer Prioritätsstufe zu erfassen.

Die Aufgaben einer Randbedingung umfassen neben Visualisierung, Erfassung und Sicherung ihrer Attribute die Überprüfung von Parameterwerten. Bei der objektorientierten Modellierung werden diese Aufgaben von entsprechenden Methoden wahrgenommen.

5.2.4 Anbindung von Randbedingungen an ein Regelobjekt

Nachdem im vorangegangenen Abschnitt die Modellierung von Randbedingungen im Mittelpunkt stand, soll nun ihre Anbindung an Regelobjekte diskutiert werden. Die dabei im Mittelpunkt stehende Frage ist die Verknüpfung mehrerer Randbedingungen mit einer Regel und ihrer Beziehungen untereinander.

Im allgemeinsten Fall kann eine im Sinne der Prädikatenlogik repräsentierte Regel als Prämisse mehrere durch logische Operatoren verknüpfte Bedingungen beinhalten. Eine bei regelbasierten Expertensystemen häufig angewandte Vorgehensweise ist, derartige Regelstrukturen als Shellskripts [HaLe90] zu erfassen und bei der Anwendung zu interpretieren. Da eine derartige Form der Regeldarstellung zum einen für den Anwender weniger Transparenz besitzt, zum anderen aber auch zusätzliche Prüf- und Interpretationsmechanismen bei der Eingabe und Anwendung von Regeln erfordert, beruht die bei MIDAS zur Anwendung kommende Vorgehensweise auf einer anderen Strategie.

Im Falle der in [Leß92] erfaßten Entwurfsregeln können logische Verknüpfungen zwischen einzelnen Randbedingungen auch in Form komplexerer, aus mehreren Einflußgrößen zusammengesetzter Entwurfsmerkmale ausgedrückt werden, denen der

Experte lediglich einen adäquaten Wertebereich zuzuweisen hat. Dadurch wird die Komplexität weg von der Eingabeschnittstelle und damit auch vom Anwender zu dem für die Merkmalsgenerierung verantwortlichen Modul hin verlagert. Zur Anbindung von Randbedingungen an ein Regelobjekt genügt in diesem Fall eine von der Regel verwaltete Listenstruktur. Eine Regel gilt dann erfüllt, wenn keine in dieser Liste erfaßten Randbedingungen verletzt wird.

Das dazugehörige objektorientierte Datenmodell ist unter Verwendung der in [CY91] gebräuchlichen Notation in Abb.23 wiedergegeben.

5.2.5 Erweiterung der Wissenserwerbskomponente

Die im vorigen Abschnitt beschriebene Erweiterung des Datenmodells um eine Klasse von Randbedingungen erfordert auch eine entsprechende Anpassung der vom Informationssystem, bzw. Akquisitionswerkzeug bereitgestellten Visualisierungs-, bzw. Eingabefunktionalität.

Hierfür wurde der Navigationsmechanismus beider Systemkomponenten um eine zusätzliche Zugriffsebene erweitert. Auf der Ebene einzelner Regelobjekte erhält der Anwender ergänzend zur vorhandenen Oberfläche eine Übersicht über alle zu einer Regel verfügbaren Randbedingungen. Durch Auswahl des dazugehörigen Listeneintrags werden die zu einer Randbedingung verfügbaren Informationen in einer eigenen Anzeige wiedergegeben.

Das Werkzeug zur Wissenserfassung ermöglicht dem Anwender auch eine Bearbeitung der angezeigten Daten.

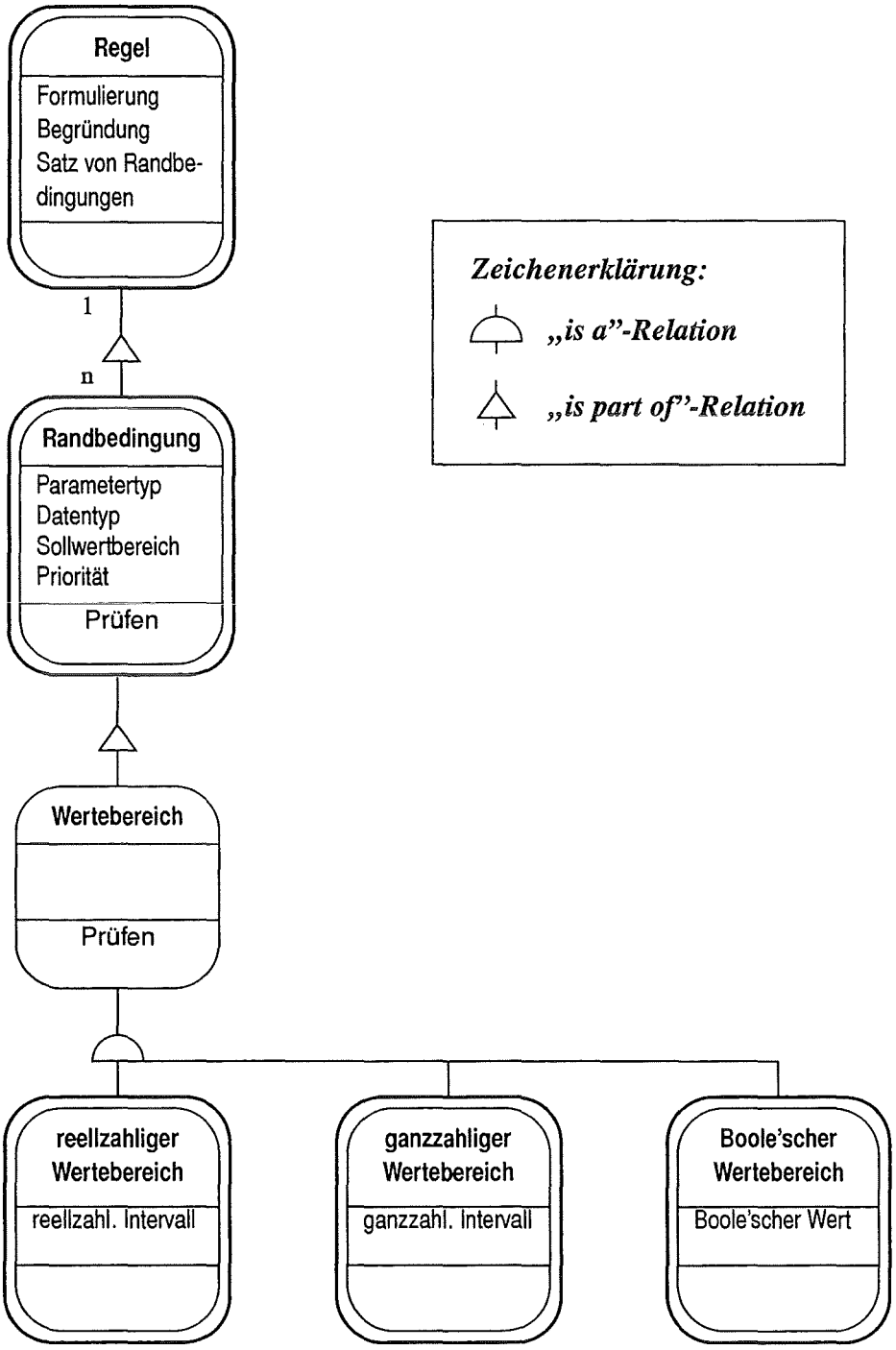


Abbildung 23: Objektorientierte Darstellung einer Regel

5.2.6 Zusammenhänge zwischen Entwurfsregeln

Mit der Modellierung einzelner Regeln sind noch nicht alle Aspekte einer Repräsentation von Regelwissen abgedeckt. Ein Gesichtspunkt, der bislang noch nicht explizit angesprochen wurde, betrifft die Beziehungen zwischen Regelobjekten.

Dem im vorangegangenen Abschnitt vorgestellten Modell entsprechend sind die einer Regel zugrundeliegenden Randbedingungen voneinander entkoppelt. Das dabei angewandte Prinzip, komplexe Strukturen von Bedingungen in entsprechende, gegebenenfalls aus mehreren Einflußgrößen zusammengesetzte Entwurfsmerkmale zu übertragen, kann auch im Falle von Entwurfsregeln angewandt werden.

Die Zugehörigkeit einer Entwurfsregel zu einem bestimmten Prozeßabschnitt wurde bereits durch die Einführung entsprechender prozeßbezogener Regelklassen bei der objektorientierten Modellierung berücksichtigt. Aufgrund dieses engen prozeßtechnischen Bezugs der Entwurfsregelklassen liegen diesen implizit dieselben Ordnungskriterien zugrunde wie der im Prozeßablauf gegebenen Folge von Verfahrensschritten. Da sich in frühen Fertigungsphasen, z.B. bei der Maskenherstellung, begangene Fertigungsfehler unter Umständen im gesamten folgenden Fertigungsprozeß bis hin zur Abformung fortpflanzen können, sollte die Einhaltung von Entwurfsregeln entsprechend der Reihenfolge ihrer zugehörigen Verfahrensschritte sichergestellt werden. Die bei einer Entwurfsdiagnose zur Anwendung kommende Regelmenge besitzt damit auf der Ebene von Regelklassen eine dem Prozeßverlauf entsprechende sequentielle Struktur (siehe Abb.24).

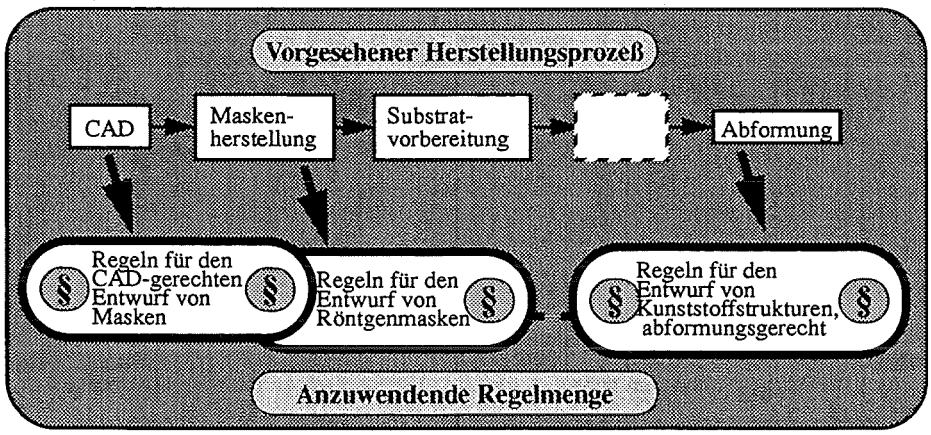


Abbildung 24: Zusammenhang zwischen Fertigungsprozeß und Regelmenge

5.2.7 Auswahl eines Regelsatzes

Welche Regeln bei einer Entwurfsdiagnose zu berücksichtigen sind, hängt beim LIGA-Verfahren entscheidend vom zu fertigenden Produkt ab, d.h. von der zugrunde-

liegenden Geometrie und dem dafür vorgesehenen Materialtyp, wie z.B. Metall oder Keramik. Mit diesen beiden Parametern liegt in den Grundzügen bereits das zur Anwendung kommende Herstellungsverfahren und damit implizit auch die anzuwendende Regelmenge fest. Diese ist auch mit fortschreitender Konkretisierung der Prozeßplanung kaum Änderungen unterworfen, da der prozeßtechnische Bezug der verschiedenen Regeln entsprechend der in [Leß92] vorgenommenen Klassifizierung auf einem vergleichsweise hohen Abstraktionsniveau der Prozeßsicht hergestellt wird.

Daher ist es sinnvoll, zunächst geeignete Regelsätze für die derzeit realisierbaren Produkttypen zu erstellen und die Auswahl eines Regelsatzes zunächst auf der Basis dieser Produkte vorzunehmen. Der Konstrukteur wählt demgemäß zuerst den für seinen Produkttyp geeigneten Regelsatz aus und nimmt danach gegebenenfalls notwendige Modifikationen an der darin enthaltenen Regelmenge vor.

5.3 Merkmalsgenerierung

5.3.1 Merkmale

Zur Durchführung einer Entwurfsdiagnose müssen abprüfbare Größen - Fakten - erzeugt werden, die vollautomatisch gegen die in einer Regel enthaltenen Referenzgrößen verglichen werden können. Ähnlich wie in der Mustererkennung kann man von sogenannten *Merkmalen* sprechen.

Definition:

> *Unter einem Merkmal versteht man eine verdichtete Information bezüglich eines Mikrostrukturobjektes, welches in der Entwurfsregel eine Rolle spielt.*

Aus semantischer Sicht werden zunächst folgende Merkmalstypen unterschieden:

- geometrische Merkmale
- prozeßbezogene Merkmale
- materialbezogene Merkmale

Beispielsweise zählen die Oberfläche oder das Volumen einer Struktur zu ihren Merkmalen. Dabei ist die Oberfläche eine komprimierte Form der Geometriedaten, die über den Entwurf in Form von Punkten und Kanten vorliegen. Bei der Datenkompression handelt es sich in diesem Falle um die Berechnung der Fläche, die algorithmisch aus den (im Geometriemodell vorhandenen) Geometrieprimitiven ermittelt werden kann. Auch die Tatsache, daß eine Struktur aus Gold besteht, bzw. daß die Ätzzeit der Struktur T_{min} beträgt, kann als linguistisches Merkmal verstanden werden, welches in diesem Falle nicht weiter komprimiert werden muß.

5.3.2 Objektorientierte Modellierung von Merkmalen

Bei der objektorientierten Modellierung von Merkmalen sind mit geometrie-, prozeß- und materialbezogenen Merkmalen semantisch unterschiedliche Merkmalstypen zu berücksichtigen. Aus diesem Grund beinhaltet das Modell für jeden dieser Merkmalstypen eine entsprechende Merkmalsklasse. Diesen typbezogenen Merkmalsklassen ist eine gemeinsame abstrakte Basisklasse übergeordnet.

Unabhängig vom semantischen Kontext enthält ein Merkmal Informationen folgender Art, die im Datenmodell durch entsprechende Attribute der Oberklasse repräsentiert werden:

- **Name des Merkmals:**

  Der Merkmalsname hat die Funktion, den semantischen Bezug des vorliegenden Merkmals in textueller Form wiederzugeben. Beispiele für derartige Merkmalsnamen sind Bezeichnungen wie „Fläche" oder „Abstand". Die hierbei verwendeten Namen korrespondieren mit den bei der Beschreibung von Entwurfsrandbedingungen benutzten Schlüsselbegriffen und stellen damit die Verbindung zu den dazugehörigen Entwurfsregeln dar.

- **Wert**

  Abhängig von ihrem semantischen Bezug können Merkmale unterschiedliche Typen von Werten beinhalten. Dabei kann es sich wie bei Flächen oder Abständen um numerische Werte handeln, denkbar sind auch andere Typen wie Boole'sche oder textuelle Werte.

- **Datentyp**

  Der Datentyp eines Merkmals geht zwar implizit aus seinem Namen hervor, d.h. dem Anwender ist im allgemeinen klar, daß Größen wie z.B. Abstände oder Flächen durch reelle Zahlen beschrieben werden, für die maschinelle Erfassung und Bearbeitung von Merkmalen muß diese Typangabe jedoch in expliziter, also für den Rechner verwertbarer Form vorliegen.

Ein weiteres Attribut von Merkmalen ist ihr Bezugsobjekt. Die Art des Bezugsobjektes hängt vom zugrundeliegenden semantischen Kontext ab. Aufgrund der dadurch gegebenen Typabhängigkeit wird dieses Attribut jeweils in den einzelnen Unterklassen spezifiziert.

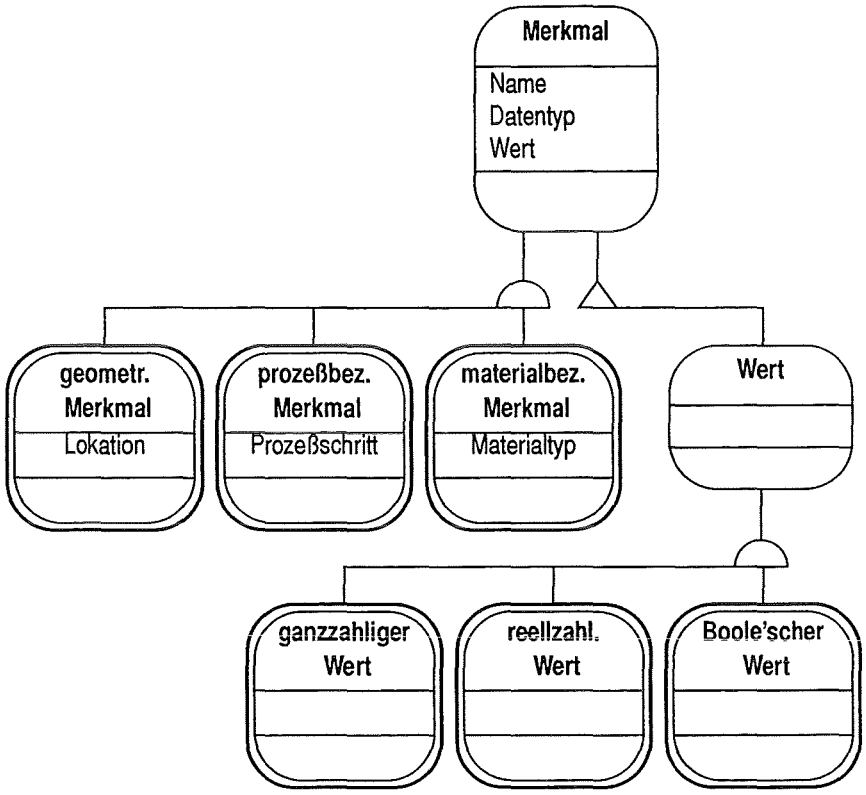


Abbildung 25: *Klassen zur Modellierung von Merkmalen*

5.3.3 Merkmalsgeneratoren

In Anlehnung an die in Abschnitt eingeführte Definition von Merkmalen, läßt sich der Begriff des Merkmalsgenerators wie folgt definieren:

Definition:

Ein Merkmalsgenerator ist ein Programm, welches aus Basisdaten „höherwertige Informationen" generiert, die als solche in den entsprechend formulierten Regeln vorkommen und die überprüft werden können.

Mit verschiedenen Strategien zur Generierung geometrischer Merkmale befassen sich die folgenden Unterabschnitte.

5.3.4 Abtastverfahren und algebraische geometrische Operationen

Ein bereits in der Mikroelektronik eingesetztes Verfahren der algorithmischen Geometrie [Brü93], das inzwischen auch im Bereich der Mikrostrukturtechnik zur Anwendung kommt [Hahn95], ist das sogenannte Abtast- oder Scan-Line-Verfahren.

Ausgangspunkt des Verfahrens ist eine in Form von Polygonen vorliegende Entwurfsgeometrie. Zuerst werden Anfangs- und Endpunkte der betrachteten Polygonkanten lexikographisch nach ihren Koordinaten geordnet. Danach bewegt man eine vertikale Linie, die sogenannte Scan-Line über die so definierte Punktmenge. Bei jeder darin enthaltenen x-Koordinate, also bei allen Anfangs-, End- und Schnittpunkten hält die Abtastlinie an. Dabei schneidet sie eine Teilmenge der zu bearbeitenden Polygonkanten. Diese werden nun sequentiell entsprechend der zugrundliegenden Problemstellung bearbeitet.

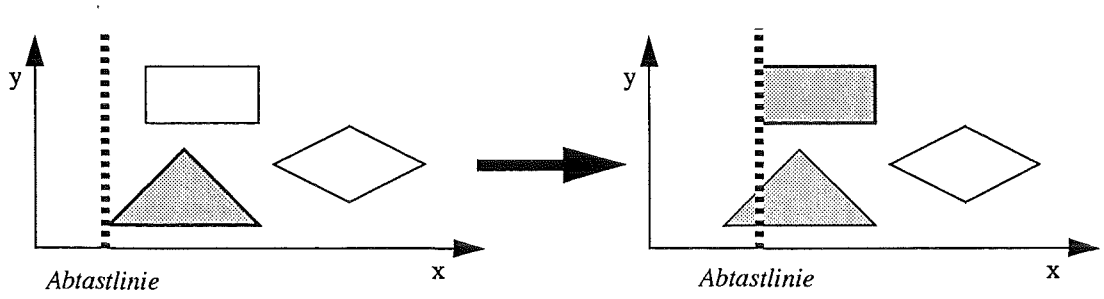


***Abbildung 26:** Grundprinzip des Abtastverfahrens*

Dabei kommen häufig sogenannte algebraische geometrische Operationen wie Boole'sche Maskenoperationen zur Anwendung. Unter einer Maske versteht man in diesem Zusammenhang die Gesamtheit aller auf einer Layoutebene liegenden Gebiete. Bei Boole'schen Maskenoperationen handelt es sich um das Äquivalent der aus der Boole'schen Algebra bekannten Operatoren AND, OR, etc., die auf Gebietsmengen angewandt werden. Einige Beispiele hierzu zeigt Abb.27.

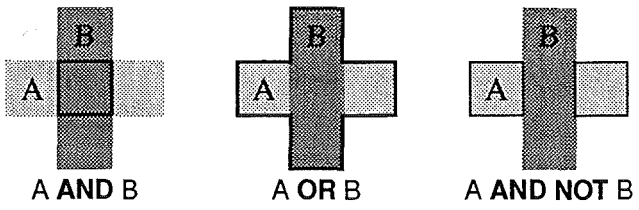


Abbildung 27: Beispiele für Boole'sche Maskenoperationen.

In der Mikroelektronik werden diese Operationen unter anderem bei der Schaltungsextraktion zur Erkennung bestimmter Bauteile wie Transistoren oder zu einer geometrischen Entwurfsregelprüfung [Brü93] verwendet. Hierbei werden durch die sequentielle Anwendung verschiedener Gebietsoperationen sogenannte Fehlermasken erzeugt. Alle in dieser Maske zu findenden Polygone weisen auf einen Entwurfsfehler hin.

Der Nachteil dieser Methode liegt vor allem im großen Speicherbedarf des Verfahrens, da unter Umständen eine große Zahl an Hilfsmasken generiert und verwaltet werden muß. Daneben weist diese Art der Entwurfsregelprüfung die Tendenz zu Pseudofehlermeldungen auf.

5.3.5 Zellbasierter Ansatz

Eine bei der Konstruktion von Mikrostrukturen aus fertigungstechnischen Gründen übliche Entwurfstechnik ist der zellbezogene Entwurf. Hierbei werden die Strukturen in Rasterfeldern auf dem Substrat angeordnet. Die Festlegung der Rasterfeldgröße richtet sich jeweils nach den Abmessungen der zu fertigenden Strukturen.

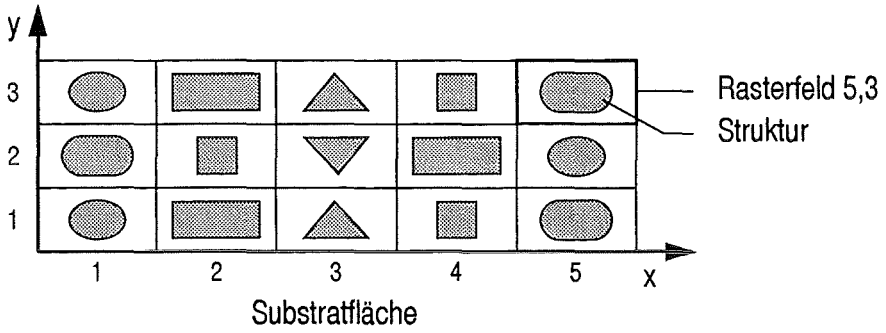


Abbildung 28: Zellbezogener Entwurfsstil

Zellbasierte Merkmalsgenerierungsstrategien [FFR95] machen sich die Tatsache zu Nutze, daß bei Einhaltung dieser Entwurfskonvention immer genau eine Figur inner-

halb einer Zelle liegt. Dies hat zum Beispiel bei Regeln, die den Mindestabstand d_{min} zweier Strukturen betreffen, den Vorteil, daß man dabei nur die Abstände der Figuren zu den Zellwänden untersuchen muß, um die Einhaltung des Mindestabstandes zu verifizieren. Ist der Abstand zu den Zellbegrenzungen bei jeder Figuren größer als $d_{min}/2$, so beträgt auch der Abstand zwischen den Strukturen mindestens d_{min}. Ohne diese Konvention hätte man hierfür zunächst ermitteln müssen, welche Strukturen sich in der Nachbarschaft der betrachteten Struktur befinden, um auf diese dann Algorithmen zur Abstandsberechnung anzuwenden.

Die Zuordnung von Merkmale zu einer Lokation innerhalb der Entwurfsgeometrie erfolgt durch die Angabe des betroffenen Rasterfeldes. Der Nachteil ist dabei, daß zur Lokalisierung einzelner Geometrieelemente wie Ecken oder Kanten zusätzliche Beschreibungsmittel zur Verfügung gestellt werden müssen.

5.3.6 Figurbasierter Ansatz

Der bei MIDAS zur Anwendung kommende figurbezogene Ansatz basiert auf einem objektorientierten Geometriemodell (GEM), das bereits bei Werkzeugen zur Entwurfsunterstützung [Egg95, ESS94] und optischen Qualitätskontrolle von Mikrostrukturen [BGH91] zum Einsatz gekommen ist. Die Ausgangsbasis dieser Applikationen bildet jeweils eine im beim Mechanikentwurf üblichen IGES-Format vorliegende CAD-Datei, die mittels eines Parsers in ein neutrales Austauschformat konvertiert wird.

5.3.6.1 Das Geometriemodell (GEM)

Mit Hilfe des objektorientierten Geometriemodells können derzeit zweidimensionale aus Geradenstücken oder Kreisbögen bestehende Figuren, dargestellt werden. Dabei besteht auch die Möglichkeit, hierarchische Strukturen aus einzelnen Figuren und Geometrieprimitiven aufzubauen. Die Klassenstruktur des Geometriemodells ist in Abb.29 gemäß der in Coad/Yourdon [CY91] üblichen Notation dargestellt. Der Übersichtlichkeit wegen wurde hierbei auf die Wiedergabe von Attributen und Methoden verzichtet.

Die oberste Stelle nimmt innerhalb der Klassenhierarchie des Geometriemodells die Klasse *Topologische Repräsentation* ein, von der alle anderen Geometrieklassen abgeleitet werden. Instanzen der Klasse *Topologische Repräsentation* enthalten als Attribute Verweise auf ein Vorgänger- und ein Nachfolgerobjekt. Auf diese Weise können geschlossene Sequenzen aus geometrischen Objekten wie z.B. Polygonzüge in Form doppelt verketteter Strukturen dargestellt werden. Ecken haben dabei jeweils eine Kante als Vorgänger, bzw. Nachfolger, Kanten werden jeweils ihre Anfangs- und Endecke als Vorgänger- und Nachfolgerobjekt zugeordnet.

Um eine einfache Vervielfältigung von Geometrieobjekten zu ermöglichen, besitzen Instanzen der Klasse *Topologische Repräsentation* außerdem noch das Attribut *Verschiebung.*

Alle komplexeren geometrischen Objekte wie Kantensequenzen oder Figuren setzen sich aus Instanzen der Klassen *Punkt* und *Kantensegment*, bzw. ihrer Unterklassen zusammen. Die Position eines Punktes wird dabei jeweils in Form von kartesischen Koordinaten festgelegt. Die Unterklassen der Klasse Punkt beinhalten noch zusätzliche geometrische oder topologische Informationen.

Ein Spezialfall von Punkten sind *Punkte auf Kantensegmenten*. Diese enthalten zusätzlich einen Verweis auf das Kantensegment, auf dem sie liegen.

Ecken werden in diesem Modell weitere topologische Informationen bezüglich der ihnen zugehörigen Kanten angegliedert. Je nach Art der angrenzenden Kanten werden Ecken den Unterklassen W-, X-, Y- und Z-Ecke zugeordnet.

Der Darstellung von Kanten dient die Klasse *Kantensegment*. Attribute eines Kantensegmentes sind die Positionen von Anfangs- und Endpunkt, sowie seine Länge und die sogenannte Bounding Box. Darunter versteht man das kleinste achsparallele Rechteck, in welches das Kantensegment einbeschrieben werden kann.

Als spezielle Formen von Kantensegmenten beinhaltet das Geometriemodell *Linien* und *Kreisbögen*. Letztere besitzen noch als zusätzliche Attribute Angaben zu Zentrum, Radius und Drehsinn des Bogenstücks.

Einen besonderen Fall von Linien stellen die sogenannten *W-Linien* dar. Diese schneiden den Rand des Bildfeldes und erfordern dadurch zusätzlich die Angabe der dazugehörigen Linie.

Geschlossene Folgen von Kantensegmenten werden von Instanzen der Klasse *Kantensequenz* repräsentiert. Neben der Abfolge von Kanten und Ecken enthalten Objekten dieser Klasse noch Informationen über die Orientierung des Kantenzuges und die Bounding Box.

Komplexere aus Kantensequenzen und untergeordneten Geometrieelementen zusammengesetzte Strukturen werden im Geometriemodell mit Hilfe der Klasse *Figur* dargestellt. Instanzen dieser Klasse enthalten Listen für Ecken, Kantensegmente, Punkte auf Kantensegmenten, Aufsetzpunkte, Kantensequenzen und auch Subfiguren. Daneben beinhalten Figuren die zu ihrer Vervielfältigung benötigten Informationen - Wiederholungsfaktoren und Verschiebungen in x- und y-Richtung - sowie Verwaltungsdaten, die Auskunft über die letzte Modifikation, die Dimension der Zahlenwerte, etc. geben. Zur Identifikation einer Figur dienen eine Kennziffer und der Figurenname.

Die Klasse *RefFigur*, beschreibt Figuren, die aus anderen Figuren mittels linearer Abbildung hergeleitet werden können. Um nicht die komplette Figureninformation speichern zu müssen, enthält eine RefFigur-Instanz die anzuwendende Abbildungsvorschrift in Form einer Transformationsmatrix und eines Translationsvektors, sowie einen Verweis auf die zu transformierende Figur.

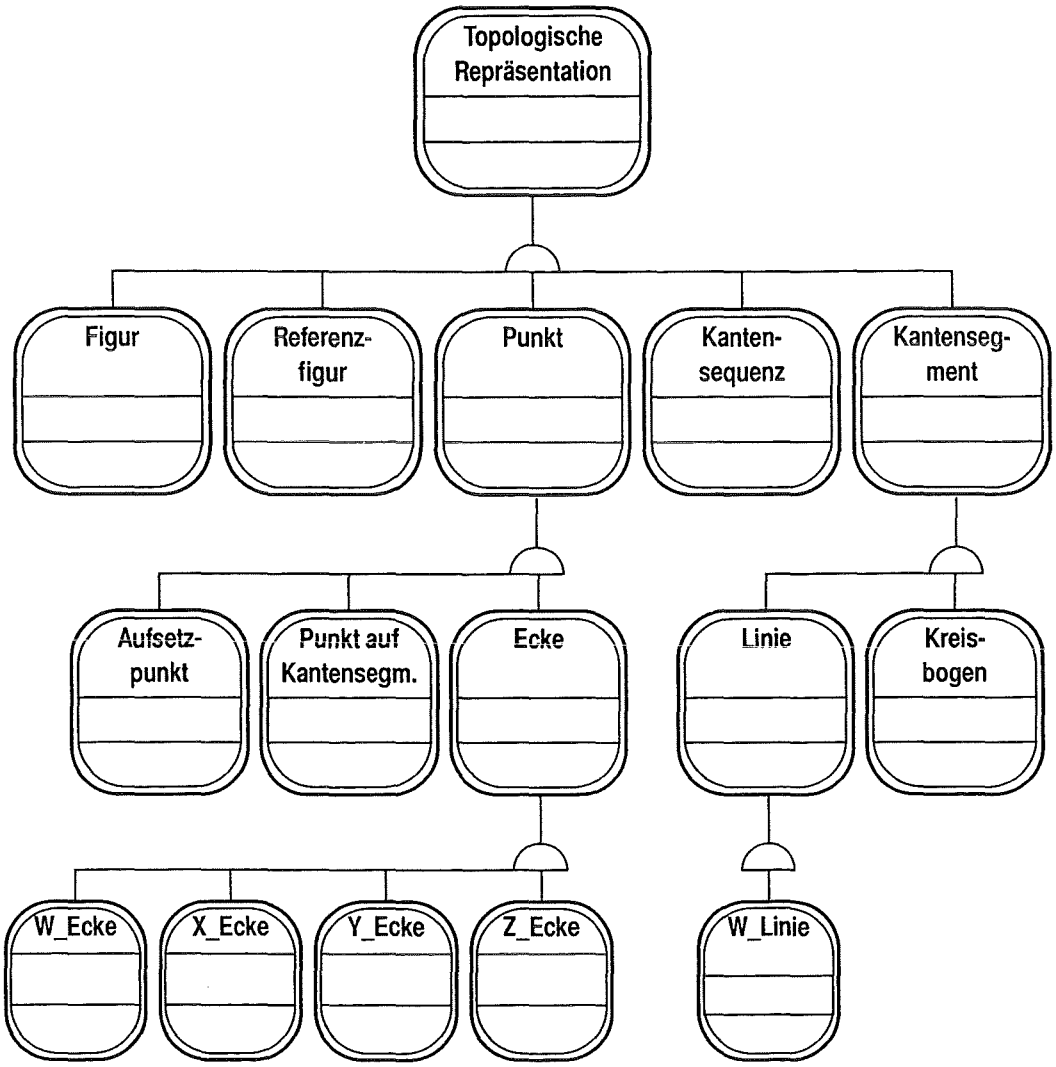


Abbildung 29: *Klassenhierarchie des Geometriemodells*

5.3.6.2 Beispiel für eine figurenbezogene Merkmalsberechnung

Die Vorgehensweise bei einer auf dem vorgestellten Geometriemodell basierenden Merkmalsberechnung soll im folgenden am Beispiel einer Flächenberechnung demonstriert werden. Das Ziel ist dabei die Berechnung der Gesamtfläche aller in einem Entwurf enthaltenen Strukturen.

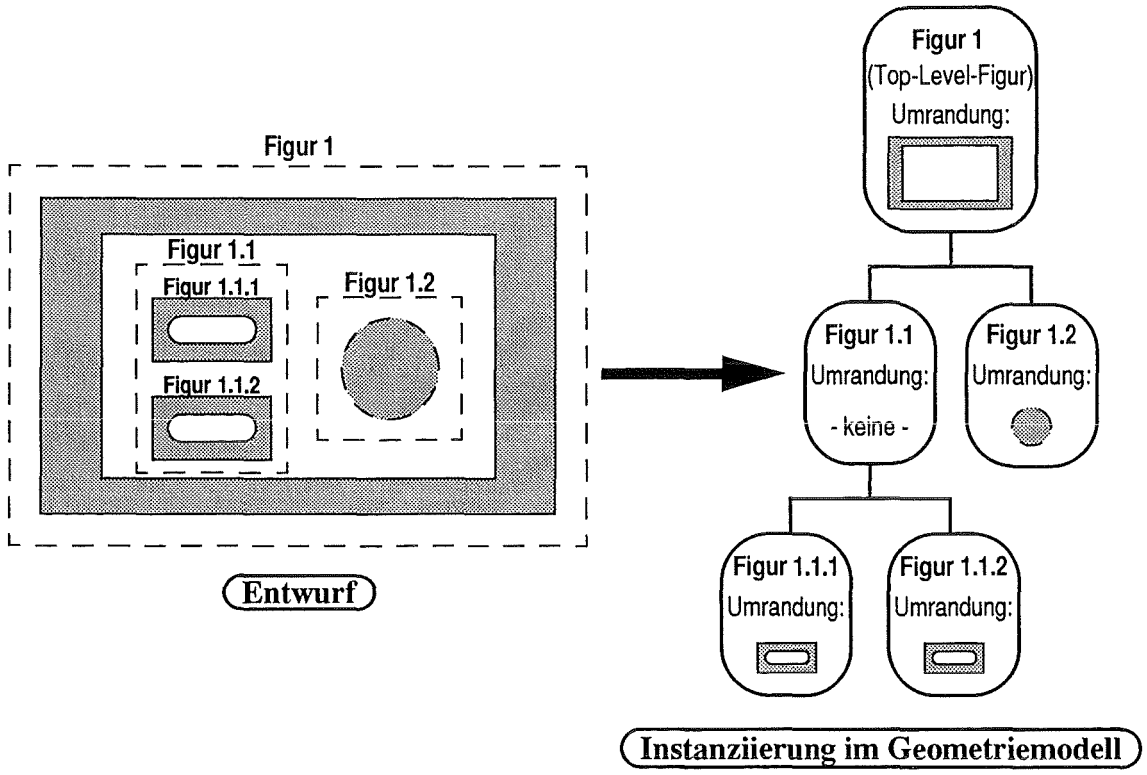


Abbildung 30: *Instanziierung einer Entwurfsgeometrie*

Sämtliche zu einer Entwurfszeichnung gehörende Informationen werden beim vorgestellten Geometriemodell in einer einzigen Instanz der Klasse Figur, der sogenannten Top-Level-Figur zusammengefaßt. Diese kann aus einer Reihe von Kantensequenzen, die ein geschlossenes Gebiet definieren, und Subfiguren bestehen. Für die Subfiguren gilt entsprechendes, auch sie können wiederum untergeordnete Figuren und Kantensequenzen beinhalten, d.h. man hat es in diesem Fall mit einer aus Figuren zusammengesetzten Baumstruktur zu tun. Ein Beispiel hierfür zeigt Abb.30.

Der Anteil einer Figur an der Gesamtfläche setzt sich aus der durch ihre eigene Umrandung gegebenen Fläche und den Flächeninhalten ihrer Subfiguren zusammen. Die Fläche der Top-Level-Figur entspricht damit der zu berechnenden Gesamtfläche.

Auf der Basis der gegebenen Baumstruktur kann man die Gesamtfläche aller Figuren mit dem in Abb.32a) dargestellten Ansatz ermitteln. Den Ausgangspunkt bildet hierbei die Top-Level-Figur. Man berechnet zunächst die durch ihre Umrandung definierte Eigenfläche und addiert dazu die Flächenanteile aller ihrer Subfiguren, die rekursiv nach demselben Prinzip bearbeitet werden. Parallel kann man beim Durchlaufen der Figurenhierarchie für jede Figur auch weitere Merkmale wie z.B. Flächenmomente oder Winkel berechnen.

Die Eigenfläche einer Figur ist durch eine Reihe geschlossener Kantensequenzen definiert. Im Geometriemodell liegen die Kantensequenzen einer Figur von außen nach innen geordnet in Form einer Liste vor. Dabei gilt die in der Mathematik übliche Konvention, daß Kantenzüge, die den Inhalt einer Struktur definieren, mathematisch positive Orientierung besitzen, während der Umrandung von Löchern ein negativer Umlaufsinn zugeordnet wird (siehe Abb.31).

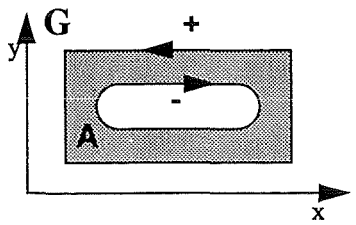


Abbildung 31: *Flächenberechnung*

Der Flächeninhalt A eines durch eine doppelpunktfreie, stückweise stetige, in Parameterform gegebenen Randkurve definierten Gebiets G ist durch folgende Gleichung gegeben [Bro82]:

$$A = \frac{1}{2} \cdot \int_{\partial G} (x dy - y dx)$$

Im vorliegenden Fall ist diese Gleichung auf alle Kantensequenzen einer Figur anzuwenden. Dafür muß der Flächenanteil jedes einzelnen Segmentes der Umrandung berechnet werden. Setzt man geeignet gewählte Parameterdarstellungen von Geradenstücken und Kreisbögen in diese Gleichung ein, erhält man die Flächenanteile dieser Kantensegmenttypen.

In Abhängigkeit von den im Geometriemodell vorhandenen Angaben erhält man folgende Ergebnisse:

(a) Der Flächenanteil A_g eines Geradenstückes mit dem Anfangspunkt $P(p_x,p_y)$ und dem Endpunkt $Q(q_x,q_y)$ beträgt:

$$A_g = \frac{1}{2} \cdot (q_x - p_x) \cdot (q_y + p_y)$$

(b) Der Flächenanteil A_k eines Kreisbogens mit Mittelpunkt $M(m_x,m_y)$, Radius R, Anfangspunkt $P(p_x,p_y)$ und Endpunkt $Q(q_x,q_y)$ beträgt:

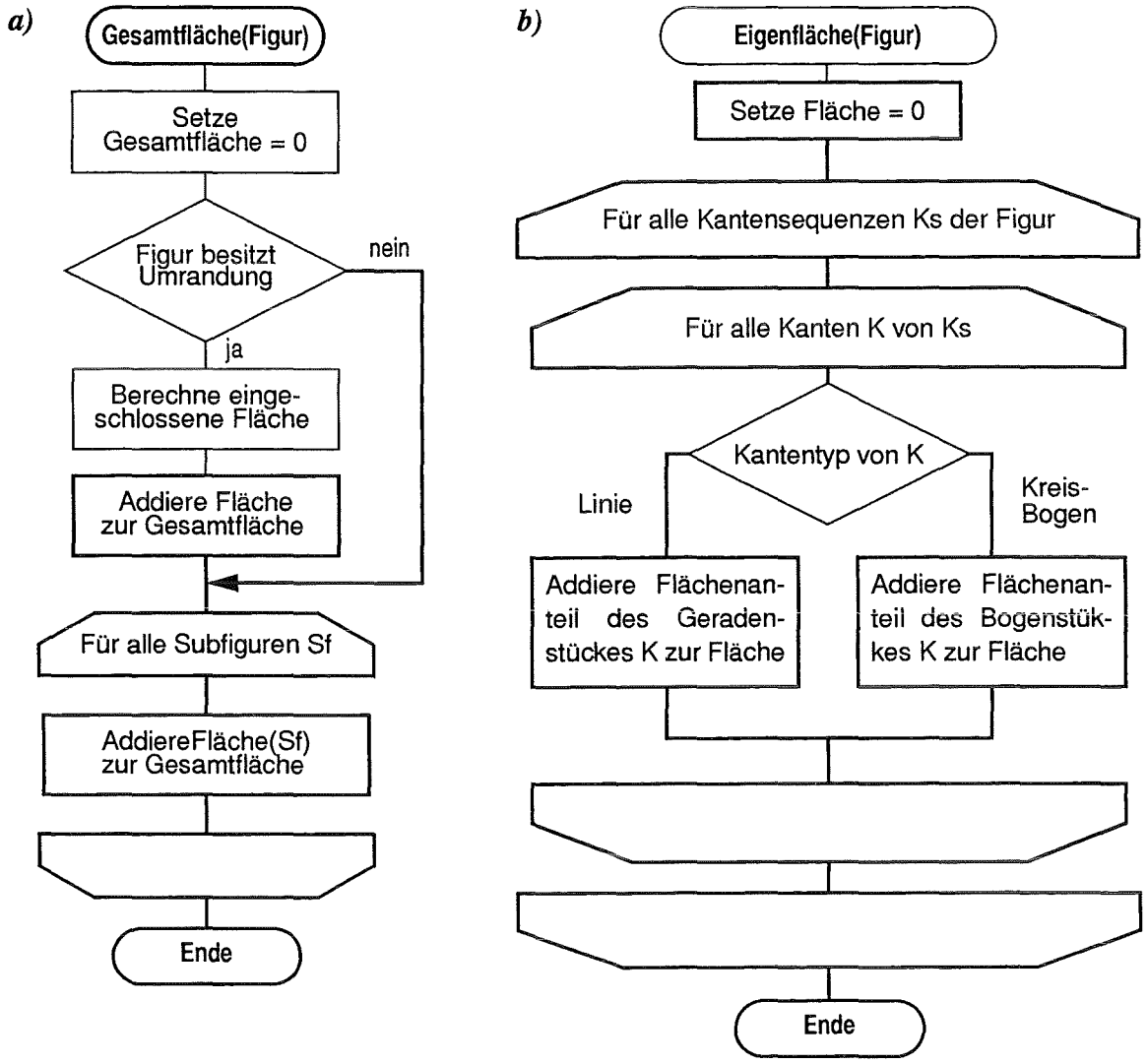
$$A_k = \frac{1}{2} \cdot (p_x - m_x) \cdot (m_y + p_y) + \frac{1}{2} \cdot (m_x - q_x) \cdot (m_y + q_y) + R^2 \cdot (\varphi_q - \varphi_p)$$

wobei $\varphi_p = \text{atan2}(p_y - m_y, p_x - m_x)$

und $\varphi_q = \text{atan2}(q_y - m_y, q_x - m_x)$

gilt.

Den hierfür entwickelten Algorithmus zeigt Abb.32b.



Algorithmus zur Berechnung der Gesamtfläche einer Figur

Algorithmus zur Berechnung des Eigenflächenanteils einer Figur

Abbildung 32: *Algorithmen zur Berechnung der Gesamtfläche eines Entwurfes*

5.3.7 Diskussion der verschiedenen Methoden zur Merkmalsgenerierung

Der Vorteil der figurbezogenen Methode liegt in der Möglichkeit, mit Hilfe eines objektorientierten Geometriemodells untergeordnete Komponenten wie einzelne Kanten oder Ecken einer Figur gezielt referenzieren zu können. Hierdurch ist bei der Diagnoseauswertung eine weitaus genauere Lokalisierung fehlerhafter Geometrieelemente möglich als bei einer rein zellbezogenen Vorgehensweise, bei der nur Rasterdaten zur Beschreibung einer Lokation verfügbar sind.

Mit den vom Geometriemodell bereitgestellten Informationen wie der Bounding Box einer Figur, können jedoch auch zellbezogene Strategien realisiert werden, die in vielen Fällen eine Vereinfachung der Algorithmen zur Merkmalsgenerierung erlauben.

Gegenüber dem Abtastverfahren hat die Verwendung dieses figurbezogenen Geometriemodells den Vorteil, daß die aus der IGES-Datei generierte Geometriebeschreibung bereits in Form einer Figurenhierarchie vorliegt, nach deren Instanziierung keine weiteren Ordnungsoperationen auszuführen sind. Das Abtastverfahren erfordert hingegen zunächst ein lexikographisches Ordnen der vorliegenden Punktemenge und bei jedem Halt der Abtastlinie zusätzliche Entscheidungen darüber, welche Operationen auf die von der Abtastlinie geschnittenen oder berührten Kanten anzuwenden sind. Die eigentliche Stärke des Abtastverfahrens liegt, kombiniert mit algebraischen Gebietsoperationen, in der Entwurfsdiagnose, bei der eine direkt mit dem CAD-System visualisierbare Fehlermaske erzeugt wird.

5.3.8 Aufbau des Moduls zur Merkmalsgenerierung

Das bei MIDAS verwendete Modul zur Merkmalsgenerierung hat die Aufgabe, für eine Diagnose relevante geometrische Parameter aus einem CAD-Entwurf zu extrahieren und in einer Ergebnisdatei abzulegen, die gleichzeitig als Eingabe für das Inferenzmodul dient.

Ansatzpunkt für die Merkmalsgenerierung ist ein objektorientiertes Geometriedatenmodell, das die geometrischen und topologischen Informationen eines Entwurfes in Form einer aus Figuren bestehenden Hierarchie bereitstellt. Die Instanziierung einer Geometrie erfolgt auf der Basis eines eigenen Dateiformats, das mit Hilfe eines Parsers aus dem im Mechanikentwurf üblichen IGES-Format generiert wird. Die Berechnung der verschiedenen Merkmale erfolgt bei MIDAS - wie im vorangegangenen Abschnitt vorgestellt - figurbezogen auf der Basis eines objektorientierten Geometriemodells.

Da derzeit verschiedenen Ortes an Werkzeugen zur Merkmalsgenerierung gearbeitet wird [FFR95], ist es unter dem Aspekt einer Einbindung dieser Werkzeuge sinnvoll, die dafür verwendeten Algorithmen von der plattformabhängigen Funktionalität der Benutzeroberfläche zu entkoppeln. Aus diesem Grund besteht das Modul zur Merkmalsgenerierung aus einer Rahmenstruktur, in welche die eigentlichen Merkmalsgeneratoren in Form eigenständiger Programme eingebunden werden.

Diese Kapselung der Algorithmen hat den Vorteil, daß die interne Programmstruktur dabei an keinerlei Voraussetzungen seitens des Systems gebunden ist. Auf diese Weise können auch Merkmalsgeneratoren, die auf anderen, z.B. zellbezogenen Strategien basieren, in das System integriert werden. Voraussetzung an diese Programme ist dabei die Einhaltung der gegebenen Austauschformate.

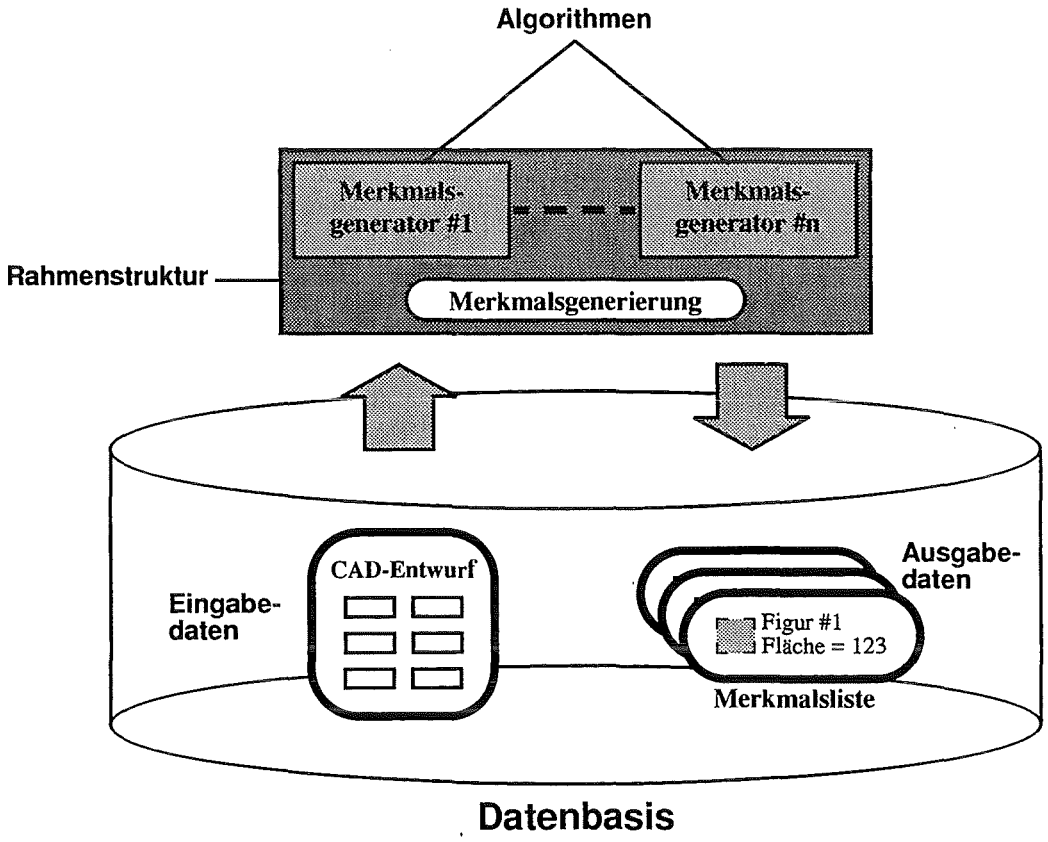


Abbildung 33: *Aufbau des Moduls zur Merkmalsgenerierung*

5.4 Inferenz

5.4.1 Grundlagen

Aufgabe einer Inferenzmaschine ist die Gewinnung neuen Wissens aus den bereits in der Wissensbasis erfaßten Inhalten und den gegebenen Falldaten. Zusammen mit der Wissensbasis bildet die Inferenzmaschine den Kern eines wissensbasierten Systems (siehe Abb.34).

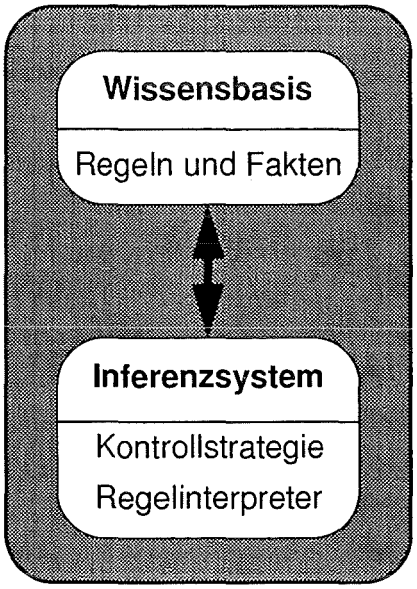


Abbildung 34: *Kern eines wissensbasierten Systems [Wat86]*

Dabei können abhängig vom Einsatzbereich des Systems und dem zu verarbeiteten Wissen jedoch unterschiedliche Inferenzstrategien zur Anwendung kommen. Bei den wichtigsten Strategien der Wissensverarbeitung handelt es sich um folgende Vorgehensweisen [ThSc89]:

- **Zielorientierte Rückwärtsverkettung**
  Bei der zielorientierten Rückwärtsverkettung werden zunächst die zu erreichenden Ziele (Goals) vorgegeben, die gegebenenfalls auch unterschiedliche Priorität besitzen können. Dabei kann es sich um direkte Benutzeranfragen oder auch um eine zu überprüfende Hypothese zur Lösungsfindung handeln. Die Ziele werden nacheinander entsprechend ihrer Gewichtung untersucht, bis die Ableitung eines Ziels aus dem vorhandenen Wissen gelingt, d.h. bis sich eine Hypothese als wahr erweist. Unbekannte Daten werden dabei je nach Anwendung auf unterschiedliche Weise, z.B. durch Fragen an den Benutzer, Datenbankzugriffe oder aber auch mit Hilfe von angeschlossenen Meßgeräten ermittelt.

- **Datenorientierte Vorwärtsverkettung**
  Die datenorientierte Vorwärtsverkettung bedient sich genau der umgekehrten, einer deduktiven Vorgehensweise. Den Ausgangspunkt dieser Strategie bilden die bekannten Fakten. Auf deren Basis erfolgt die Ableitung neuer Fakten, die dem Benutzer als Ergebnis vorgelegt werden können.

- **Inhaltsorientierte Wissensverarbeitung**
  Im Falle einer inhaltsorientierten Wissensverarbeitung gewinnt man aus den bekannten Fakten die jeweils als nächstes anzuwendenden Suchkriterien, d.h. die Zwischenergebnisse einer Suche können unter Umständen auch eine Änderung der Suchstrategie bewirken. Dies kann gegebenenfalls zum Entfernen unzweckmäßiger Regeln oder zur Neuformulierung von Regeln führen

5.4.2 Die angewandte Inferenzstrategie

Bei MIDAS ist die Inferenzkomponente für die Durchführung der Entwurfsregelprüfung verantwortlich. Die Arbeitsweise dieses Programmes ist im wesentlichen von der Struktur der Regelmenge vorgegeben. Durch die bei der Modellierung von Regelwissen und Merkmalen geleistete Vorarbeit kann bei der Inferenzkomponente ein vergleichsweise einfaches, sequentielles Prüfverfahren zur Anwendung kommen.

Bei der hierbei eingesetzten Inferenzstrategie handelt es sich um eine datenorientierte Vorwärtsverkettung. Das zu bearbeitende Faktenwissen liegt in Form von geometrischen Merkmalen vor, auf die ein geeignet gewählter Regelsatz angewandt wird.

Die Abarbeitung der in der Konstruktionswissensbasis enthaltenen Regeln erfolgt klassenweise in der Reihenfolge der dazugehörigen Prozeßabschnitte. Ob sich eine Regel, bzw. eine der in ihr enthaltenen Bedingungen auf ein Merkmal bezieht, wird dabei jeweils anhand des Merkmalsidentifikators überprüft.

Falls eine Regel zur Anwendung kommt, nimmt die Inferenzkomponente einen Vergleich des Merkmalswertes mit dem in der Regel vorgegebenen Sollwertbereich vor. Bei einer Verletzung des Wertebereichs erfolgt ein entsprechender Eintrag in ein Fehlerprotokoll. Darin werden die Kenndaten des betroffenen Merkmals - Merkmalstyp, Wert und Lokation - sowie die im Zusammenhang mit diesem Merkmal verletzten Regeln erfaßt.

5.5 Interpretation der Diagnoseresultate

Den Abschluß einer Entwurfsdiagnose bildet die Interpretation der Diagnoseresultate im Dialog mit dem Konstrukteur. Dieser muß zunächst über alle potentiellen Schwachstellen seines Entwurfes informiert werden, um dann im Einzelfall darüber entscheiden zu können, ob ein Konstruktionselement zu korrigieren ist. Er benötigt hierzu Informationen darüber, worin die Regelverletzung besteht und welche Auswirkungen sie auf die Fertigung hat. Ebenso sind in diesem Zusammenhang mögliche

Korrekturmaßnahmen von Bedeutung.

5.5.1 Modellierung von Fehlermeldungen

Zur Darstellung der bei der Entwurfsdiagnose festgestellten Konstruktionsfehler wurde als Schnittstelle zwischen der Inferenzkomponente und dem Modul zur Fehlervisualisierung eine Klasse namens „Fehlerprotokoll" eingeführt. Hierbei handelt es sich um eine Liste von Fehlermeldungen. Diese werden im dafür entwickelten Modell durch Instanzen einer entsprechenden Klasse „Fehlermeldung" repräsentiert. Bei den Attributen einer Fehlermeldung handelt es sich um:

- **das betroffene Merkmal**
  Dieses Attribut beinhaltet alle spezifischen Kenndaten des fehlerhaften Konstruktionselementes - Merkmalstyp, Datentyp und Wert, sowie im Falle geometrischer Merkmalen dessen Lokation.
- **eine Liste mit den verletzten Regeln**
  In der Regelliste werden alle im Zusammenhang mit dem gegebenen Merkmal verletzten Gestaltungsregeln erfaßt. Diese beinhalten in der in Abb.7 vorgestellten Form alle vom Konstrukteur benötigten Informationen über die Art der Regelverletzung, den prozeßtechnischen Hintergrund und graphische Beispiele für eine ungünstige, bzw. empfehlenswerte Gestaltung eines Entwurfes.

Die wesentliche Aufgabe der Klassen „Fehlerprotokoll" und „Fehlermeldung" ist die visuelle Darstellung ihrer Inhalte. Dementsprechend stellen beide Klassen eine Visualisierungsmethode „Anzeige" zur Verfügung. Außerdem beinhalten die Klassen die Methoden „Lesen" und „Schreiben" für den Zugriff auf die Ein-, bzw. Ausgabeströme.

Das zur Repräsentation von Fehlermeldungen entwickelte Modell zeigt Abb.35. Die Darstellung basiert auf der in [CY91] üblichen Notation.

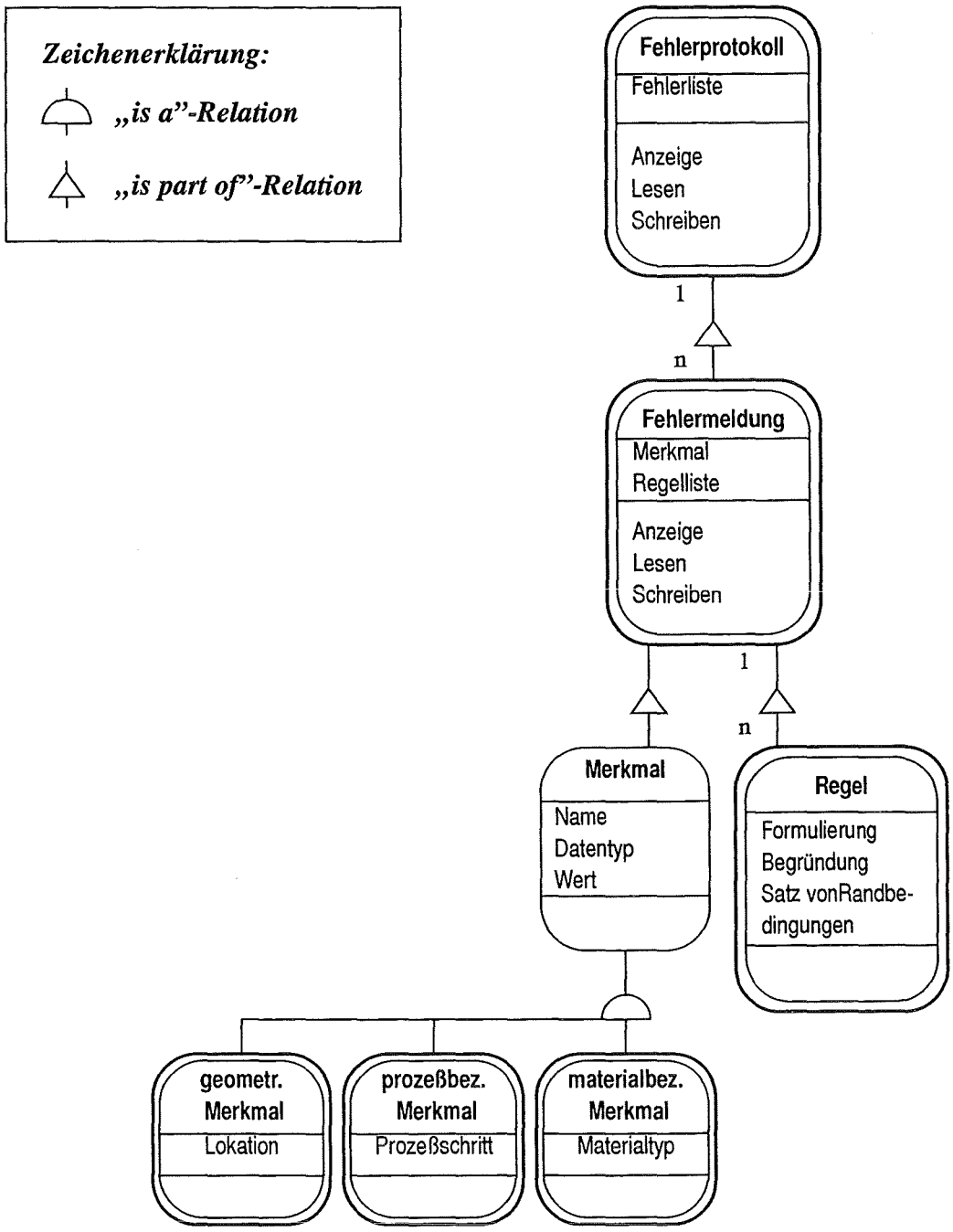


Abbildung 35: Objektorientierte Darstellung eines Fehlerprotokolles

5.5.2 Aufbau des Moduls zur Fehlervisualisierung

Mit Hilfe des Moduls zur Fehlervisualisierung soll der Konstrukteur darüber informiert werden *wo* sich Schwachstellen in seinem Entwurf befinden und *welcher Art* sie sind. Entsprechend dieser Hauptfunktionen besteht dieses Subsystem aus zwei Teilkomponenten - der Anzeige des Fehlerprotokolls und einer graphischen Darstellung der Entwurfsgeometrie.

Die Bildschirmausgabe des Fehlerprotokolles greift auf die Visualisierungsmethoden der Klassen „Fehlerprotokoll" und „Fehlermeldung" zurück. Das Fehlerprotokoll wird dabei in Form einer Auswahlliste dargestellt, die alle ermittelten Konstruktionsfehler enthält. Die Listeneinträge setzen sich aus dem *Typ* des betroffenen Geometriemerkmals und dessen *Lokation* zusammen. Da der Konstrukteur in vielen Fällen nur an bestimmten Teilmengen der angezeigten Fehlermeldungen interessiert ist, enthält das System zusätzliche Kontrollelemente die eine entsprechende Filterung der Ausgabe erlauben. Auswahlkriterien sind dabei die *Priorität* der verletzten Regeln, der *Merkmalstyp* und die *Lokation* der Fehler innerhalb der Entwurfsgeometrie.

Einzelne Fehlermeldungen können durch Selektion des dazugehörigen Listeneintrages abgerufen werden. Die hierbei aktivierte Bildschirmanzeige enthält die Kenndaten des betroffenen Merkmals und eine Auswahlliste mit den im Zusammenhang mit diesem Merkmal verletzten Gestaltungsregeln. Deren Bildschirmanzeige kann wiederum durch Auswahl des entsprechenden Listeneintrags aktiviert werden.

Alle Fehler, die einer konkreten Lokation innerhalb der Entwurfsgeometrie zugeordnet werden können, werden außerdem noch in einer graphischen Form wiedergegeben. Hierbei greift MIDAS auf die Funktionalität eines für die formale Prüfung von CAD-Entwürfen entwickelten Visualisierungswerkzeuges zurück [Egg95], das die Umsetzung einer im Geometriemodell instanziierten Figurenhierarchie in eine graphische Anzeige ermöglicht. Fehlerhafte Geometrieelemente werden in dieser Übersicht farblich hervorgehoben. Dabei hat der Anwender außerdem die Möglichkeit, sich mit Hilfe zusätzlicher Kontrollelemente gezielt über bestimmte Teilaspekte seines Entwurfes zu informieren.

Hierzu gehören:

- Ein sogenannter Panner, mit dessen Hilfe eine Bewegung in der Zeichnung möglich ist.
- Schalter zum Zoomen, mit denen Bildausschnitte vergrößert oder verkleinert werden können.
- Schalter zum Anzeigen bzw. Ausblenden aller Fehler.

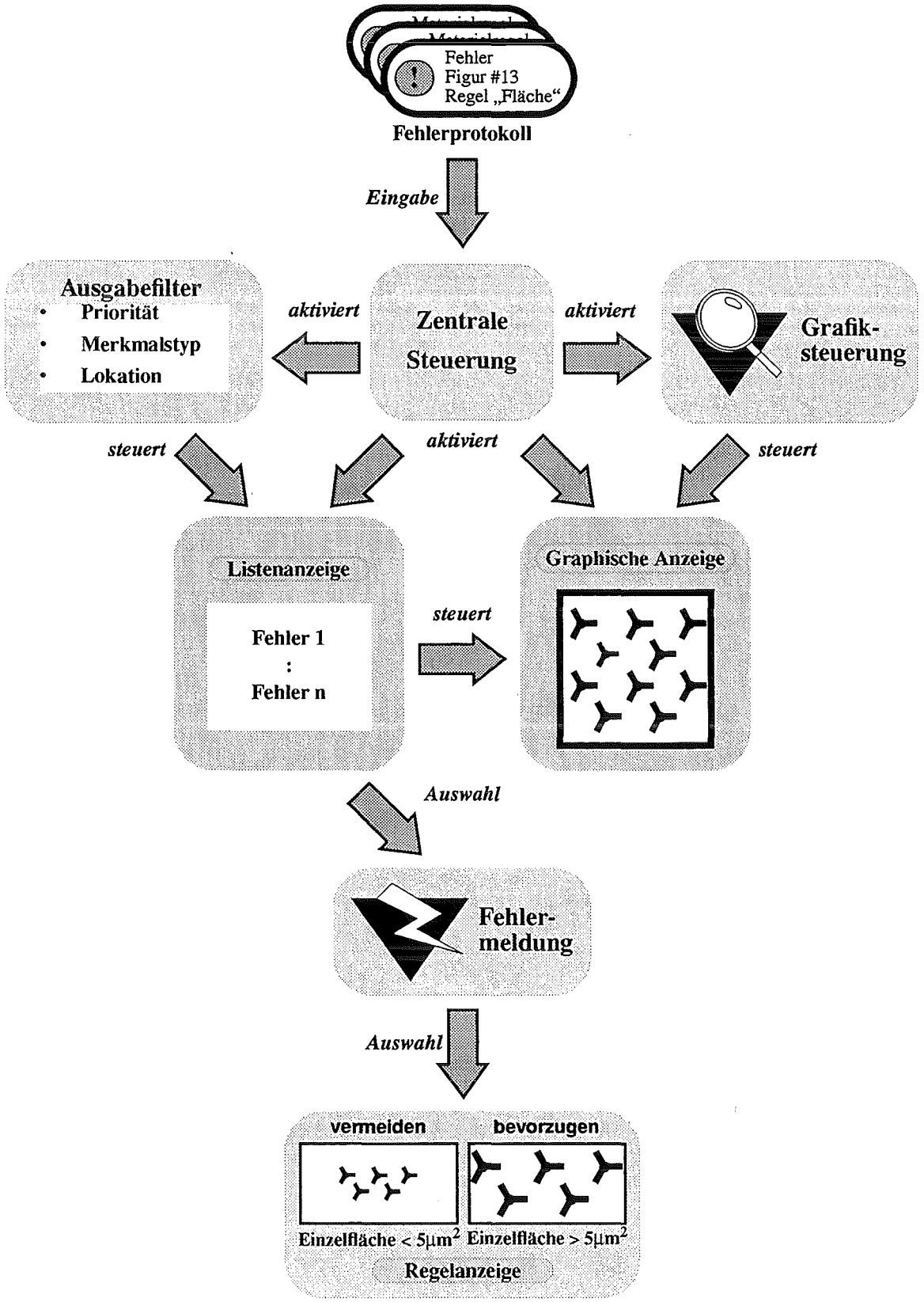


Abbildung 36: Aufbau des Moduls zur Fehlervisualisierung

6. Der Ablauf einer Entwurfsdiagnose am Beispiel eines Planetengetriebes

Zur Veranschaulichung des vorgestellten Diagnosekonzeptes soll im folgenden Ablauf einer Entwurfsdiagnose in allen Phasen am Beispiel eines realen Layouts demonstriert werden.

Bei dem in Abb.37 dargestellten Entwurf handelt es sich um das Layout eines Planetengetriebes, das am Institut für Mikrostrukturtechnik des Forschungszentrums Karlsruhe im Kundenauftrag gefertigt wird.

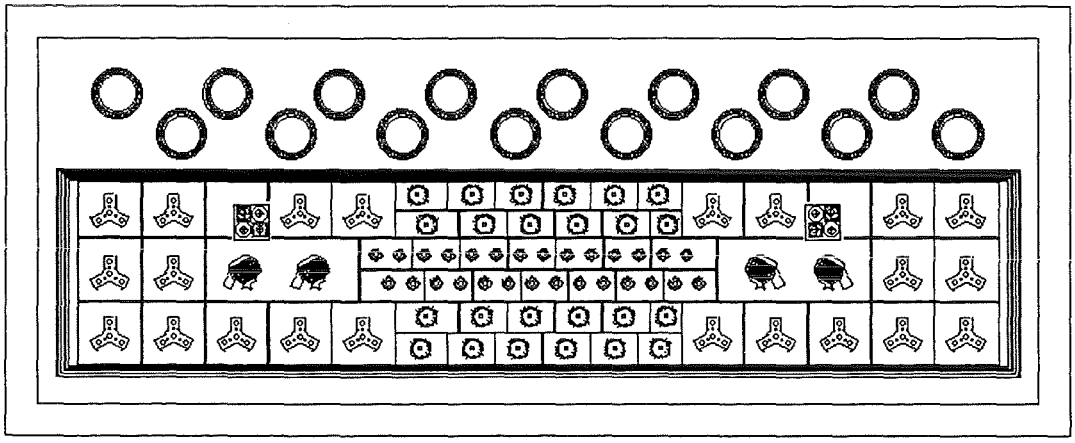


Abbildung 37: Layout eines Planetengetriebes

6.1 Der zur Anwendung kommende Regelsatz

Für dieses Anwendungsbeispiel wurde ein aus flächenbezogenen Entwurfsregeln bestehender Regelsatz zusammengestellt. Diese Wahl hat den Vorteil, daß den Regeln mit flächenbezogenen Größen eine vergleichsweise anschauliche Form von Geometriemerkmalen zugrundeliegt, deren Überprüfung jedoch in der Praxis nicht trivial ist.

Im einzelnen handelt es sich hierbei um folgende Regeln:

Regel 1: Einzelfläche

Die Mindestgröße einer Einzelstruktur auf dem Substrat sollte 5 μm^2 überschreiten.

Begründung:

Bei der Durchführung des Galvanikschrittes muß eine minimale Haftung der Strukturen gewährleistet sein. Hierzu muß eine Struktur über diese Mindestfläche verfügen.

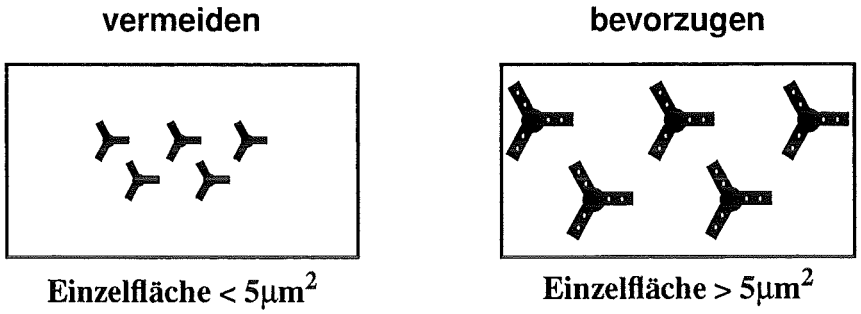


Abbildung 38: Beispiele zu Regel 1

Regel 2: Flächenausnutzung

Die Flächenbelegung eines Substrates sollte mehr als 80% betragen.

Begründung:

Aus Kostengründen sollte die Substratfläche möglichst gut ausgenutzt werden.

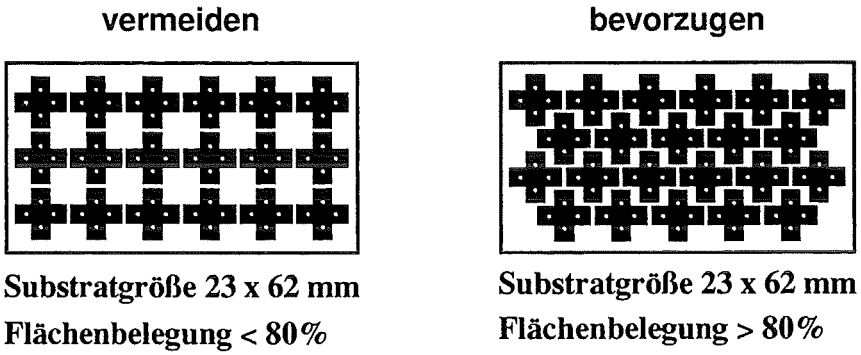


Abbildung 39: Beispiele zu Regel 2

Regel 3: Gesamtfläche

Die Summe der zu galvanisierenden Fläche muß mindestens 1000 µm² betragen.

Begründung:

Für eine gleichmäßige Beschichtung ist diese Mindestfläche erforderlich.

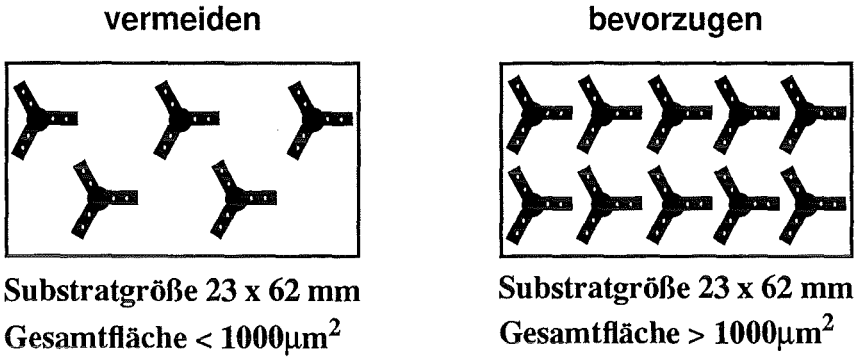


Abbildung 40: Beispiele zu Regel 3

Logisch sind alle drei Regeln durch den UND-Operator verknüpft, d.h. sie werden von der Inferenzmaschine gemeinsam abgearbeitet.

6.2 Erfassung der Regeln

Die Erfassung der vorgestellten Entwurfsregeln soll in diesem Abschnitt am Beispiel von Regel 1 demonstriert werden. Die hierfür verwendete Wissenserwerbskomponente wird über das in Abb.41 dargestellte Startfenster von MIDAS aktiviert.

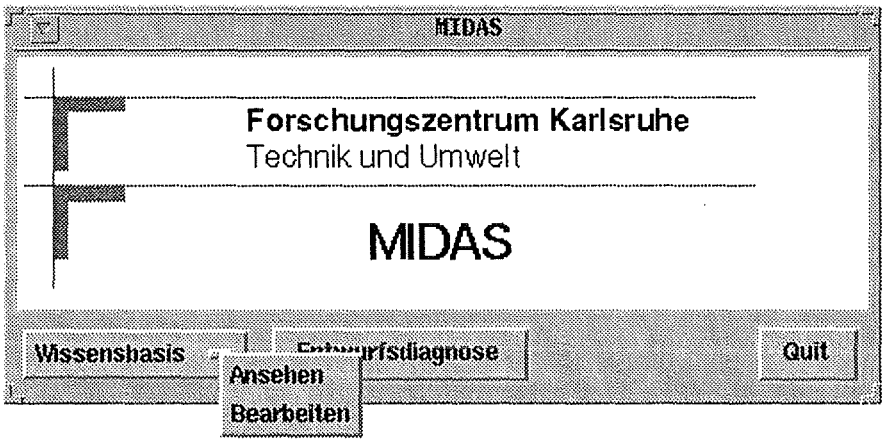


Abbildung 41: Das Startfenster von MIDAS

Informations- und Wissenserwerbskomponente werden dabei über die Optionen „ansehen", bzw. „bearbeiten" des Schalters „Wissensbasis" aufgerufen; der Schalter „Entwurfsdiagnose" dient der Aktivierung des Diagnosesystems.

Über das in Abb.42 dargestellte Kontrollfenster der Wissenserwerbskomponente hat

der Benutzer die Möglichkeit, in Form eines Menüs auf eine Wissensbasis zuzugreifen. Zum Einlesen einer bestehenden Wissensbasis dient die Option „öffnen" bei der ein Werkzeug zur Navigation innerhalb des UNIX-Dateisystems aufgeblendet wird.

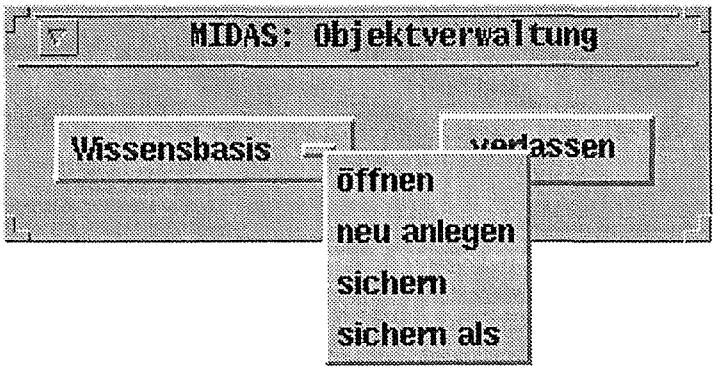


Abbildung 42: *Kontrollfenster der Wissenserwerbskomponente*

Nach Auswahl der zu bearbeitenden Wissensbasis erhält der Benutzer eine listenartige Übersicht über alle verfügbaren Klassen von Fertigungswissen (Abb.43). Der Inhalt der Klassenliste kann mit Hilfe zusätzlicher Kontrollelemente weiterbearbeitet werden. Der Zugriff auf eine dieser Klassen erfolgt durch Selektion des entsprechenden Listeneintrages.

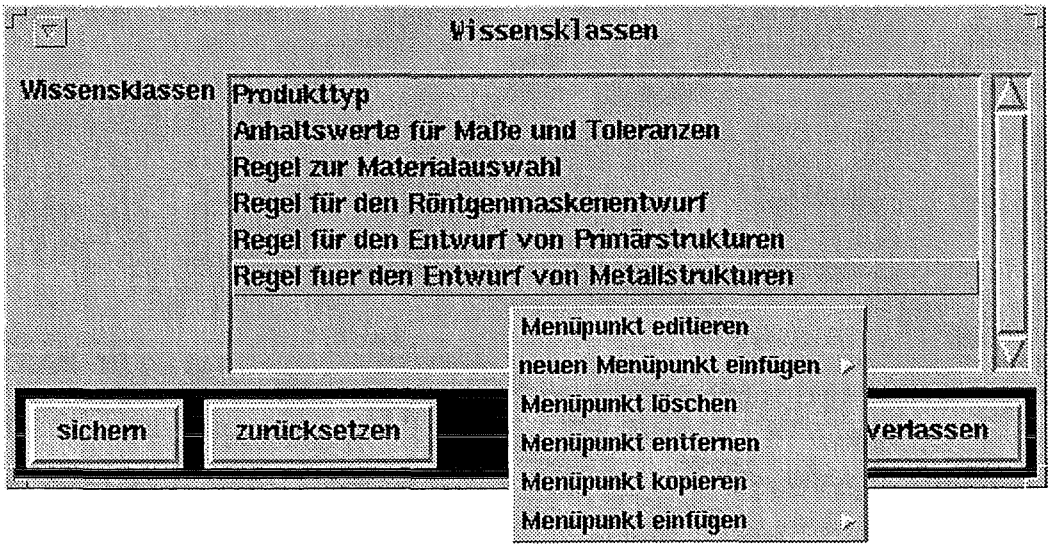


Abbildung 43: *Bildschirmfenster zur Auswahl einer Wissensklasse*

Hierbei wird ein Bildschirmformular aktiviert, das die zur selektierten Klasse verfügbaren Informationen beinhaltet. Im Falle der bei der Galvanoformung zur Anwendung kommenden Beispielregel ist hier die Klasse „Entwurfsregel für Metallstrukturen" zu selektieren. Dort werden dem Benutzer alle vorhandenen Instanzen dieser Regelklasse angezeigt. Änderungen der Instanzenliste sind mit Hilfe zusätzlicher Kontrollelemente möglich.

Zur Bearbeitung eines Objektes steht dem Benutzer ein Bildschirmformular zur Verfügung, das dessen Namen und Attribute beinhaltet (siehe Abb.44). Bei den Attributwerten kann es sich um ganze und reelle Zahlen, Zeichenketten, Texte oder Bitmapdateien im PBM-Format (Portable Bitmap) handeln.

Im vorliegenden Fall beinhaltet das Formular Texteingabefelder für den Regelnamen, die Regelformulierung und die Begründung der Regel, sowie Anzeigefelder für die graphischen Attribute „vermeiden" und „bevorzugen"

Sämtliche Eingabewerte werden bei Sicherungsoperationen oder beim Schließen des Formulars mit den zulässigen Wertebereichen verglichen. Bei Fehleingaben, bzw. wenn das Formular zum Zeitpunkt des Verlassens noch ungesicherte Eingaben enthält ergeht eine entsprechende Meldung an den Benutzer.

Die numerisch zu überprüfenden Merkmalsgrößen und die für sie geltenden Richtgrößen werden mit Hilfe des Attributes „Randbedingungen" erfaßt. Hierfür enthält das Eingabeformular eine Liste, über die Randbedingungen für den Entwurf definiert oder modifiziert werden können. Die Bearbeitung der Liste wird - wie bereits bei der Klassen- bzw. Objektauswahl beschrieben - über zusätzliche Kontrollelemente abgewickelt, welche die hierfür benötigten Operationen bereitstellen.

Die Eingabe einer Entwurfsrandbedingung erfolgt über deren eigenes Bildschirmformular, das bei Selektion des entsprechenden Listeneintrages aufgeblendet wird. Dieses Formular enthält Anzeigefelder für den Namen des betroffenen Konstruktionselementes, sowie Datentyp, Maßeinheit und zulässigen Wertebereich des Entwurfsparameters.

Daneben kann der Benutzer einer Entwurfsrandbedingung außerdem eine Prioritätsstufe zuweisen. Der Wertebereich dieses Attributes besteht aus den fünf diskreten linguistischen Werten „notwendig", „wichtig", „immer von Vorteil", „häufig von Vorteil" und „gelegentlich von Vorteil".

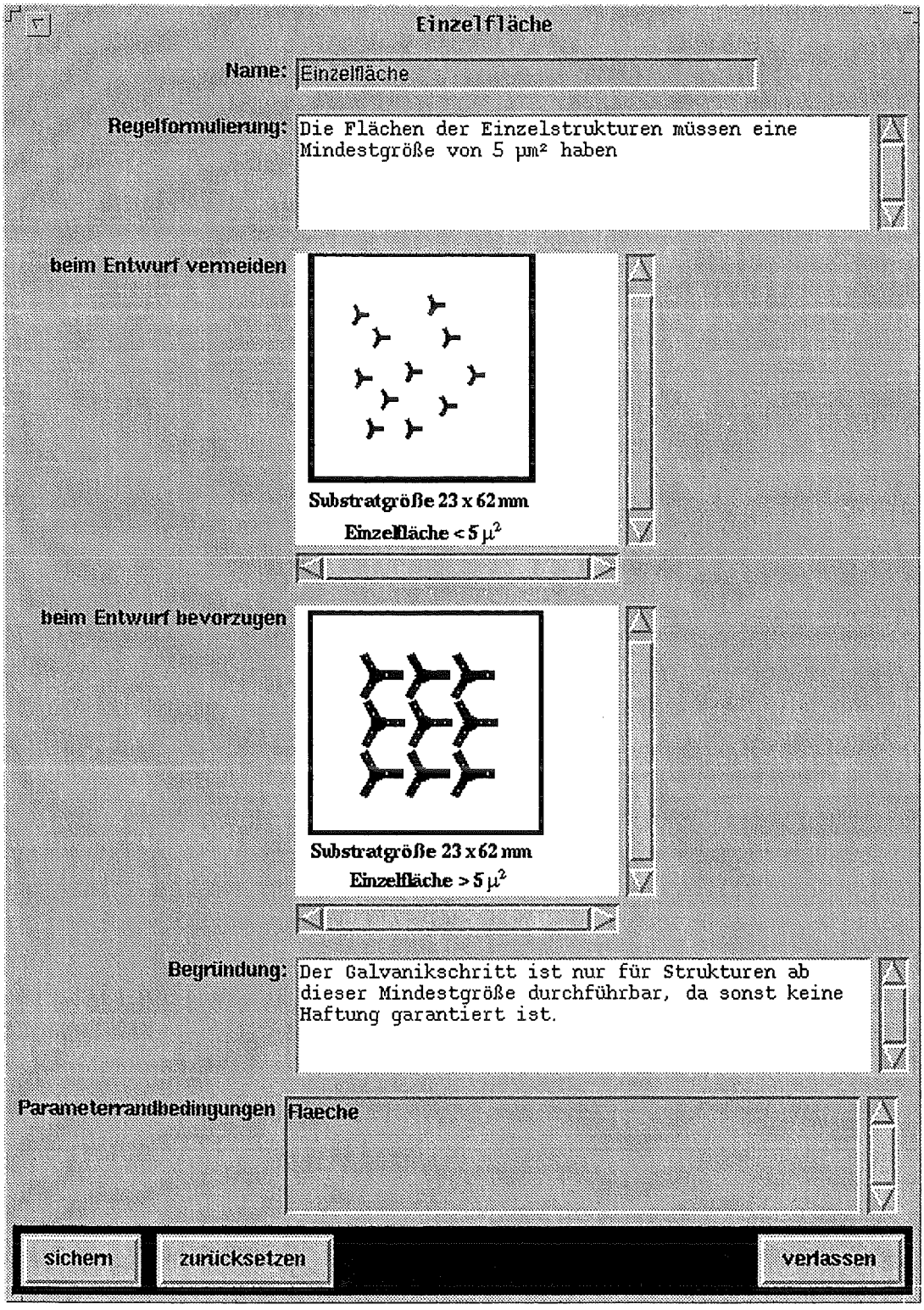


Abbildung 44: Darstellung von Regel 1

Die in Regel 1 enthaltene Randbedingung (Abb.44) bezieht sich auf die Grundfläche einer Mikrostruktur. Hierbei handelt es sich um eine numerische Größe vom Datentyp Fließkommazahl, die in der Maßeinheit μm^2 angegeben wird. Der Mindestwert für dieses Merkmals beträgt $5\mu m^2$, die Obergrenze liegt bei $1{,}4 \cdot 10^9\ \mu m^2$, was in etwa der verfügbaren Substratfläche entspricht. Aus fertigungstechnischer Sicht ist die Einhaltung dieser Randbedingung notwendig.

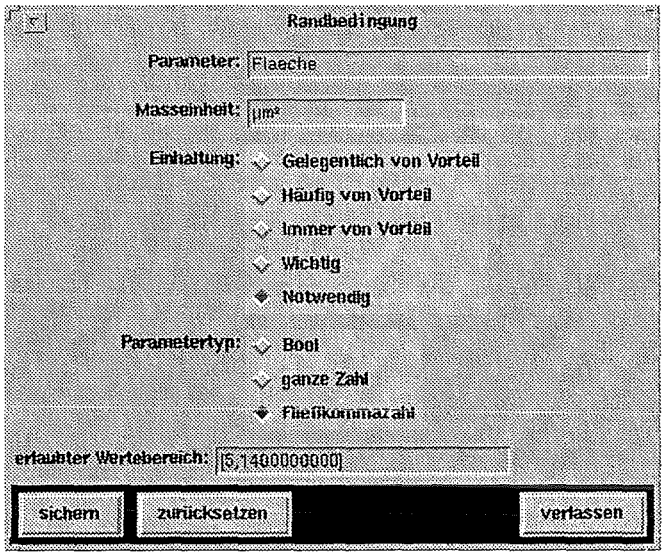


Abbildung 45: Darstellung von Regel 1 als Entwurfsrandbedingung

6.3 Durchführung der Entwurfsdiagnose

Das Diagnosesystem wird ebenso wie Wissenserwerbs- und Informationskomponente über das Startfenster von MIDAS aktiviert. Die in Abb.46 wiedergegebene Oberfläche dieser Systemkomponente beinhaltet Kontrollelemente zur Regelselektion, Merkmalsgenerierung, Diagnosedurchführung und Aktivierung der Fehleranzeige.

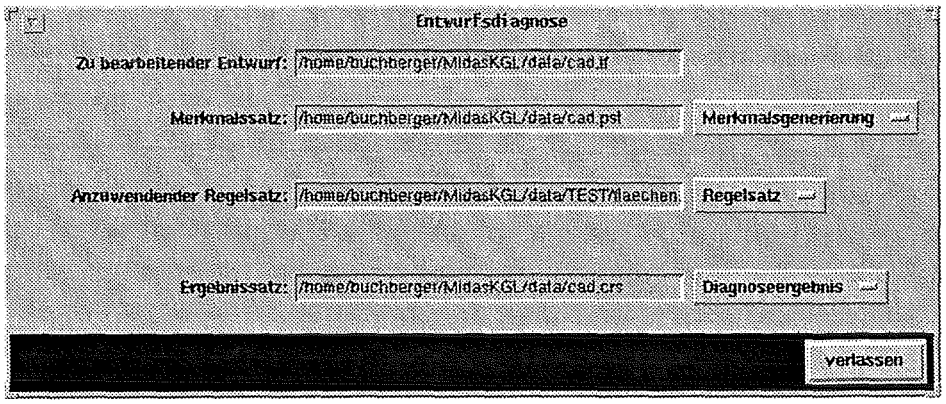


Abbildung 46: Kontrollfenster der Diagnosekomponente

6.3.1 Der zu prüfende Entwurf

Den Namen der zu bearbeitenden Entwurfsdatei kann der Anwender entweder über das entsprechende Eingabefeld des Kontrollfensters oder das zum Schalter „Merkmalsgenerierung" gehörende Menü vorgeben.

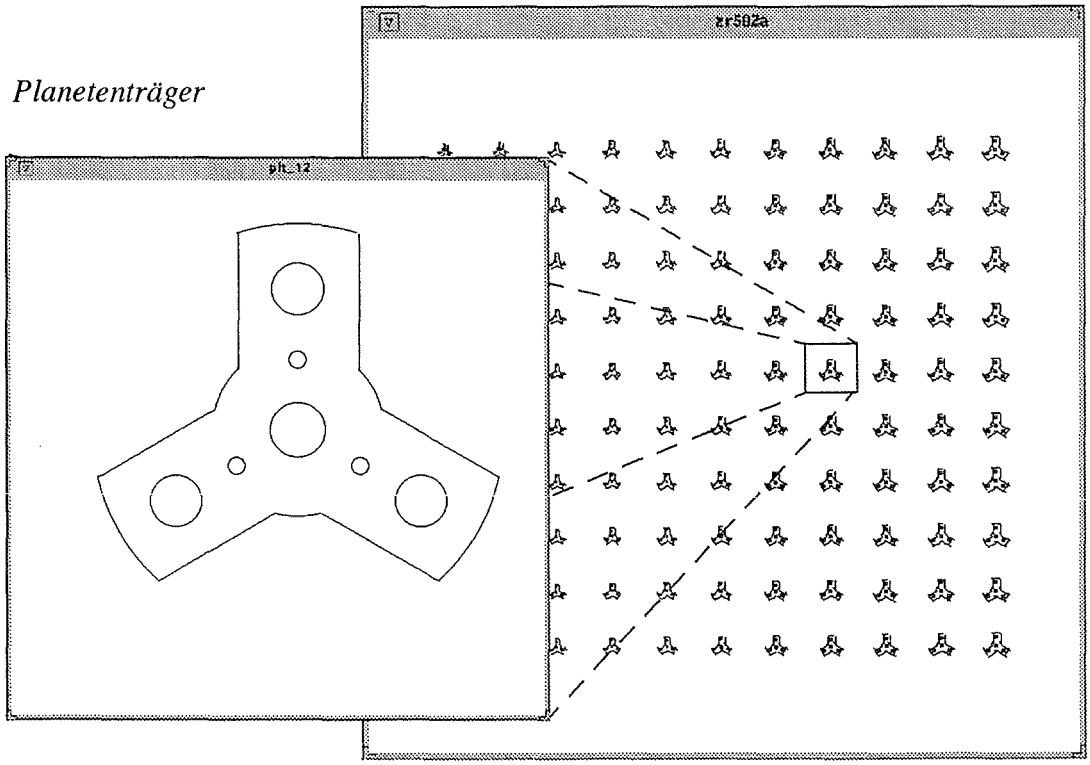


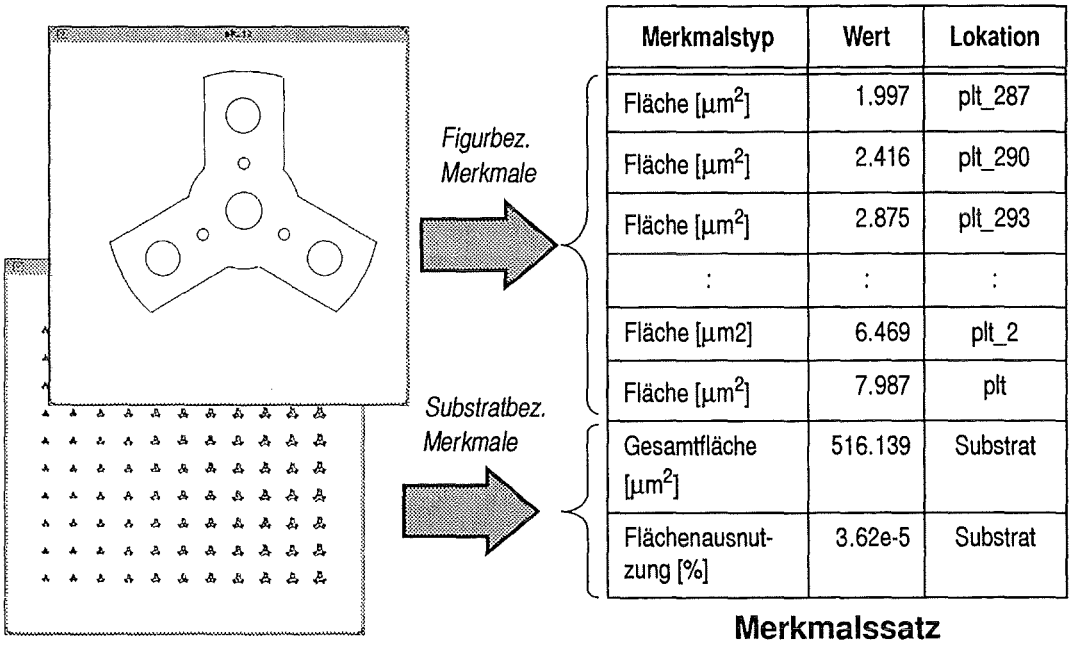
***Abbildung 47:** Beispielentwurf und Ausschnittsvergrößerung*

Für die Anwendung der vorgestellten Gestaltungsregeln wurde exemplarisch die in Abb.47 dargestellten Trägerstrukturen des Planetengetriebes herausgegriffen, die hierzu in verschiedenen Größenstufen konstruiert wurden. Ihre Fläche liegt dabei in der Nähe des für Regel 1 kritischen Wertes von $5\mu m^2$.

Das gezeigte Entwurfszeichnung enthält ein Array von 10 x 11 dieser Strukturen.

6.3.2 Merkmalsgenerierung

Das Modul zur Merkmalsgenerierung wird über den entsprechenden Schalter der in Abb.46 dargestellten Kontrolloberfläche aktiviert. Daraufhin werden die Werte der in den Entwurfsregeln genannten Merkmale aus der Entwurfszeichnung extrahiert und in einer Datei gesichert, deren Name im Texteingabefeld „Merkmalssatz:" angezeigt wird. Bereits vorhandene Merkmalssätze können über dieses Eingabefeld auch eingelesen werden.



Merkmalstyp	Wert	Lokation
Fläche [µm²]	1.997	plt_287
Fläche [µm²]	2.416	plt_290
Fläche [µm²]	2.875	plt_293
:	:	:
Fläche [µm2]	6.469	plt_2
Fläche [µm²]	7.987	plt
Gesamtfläche [µm²]	516.139	Substrat
Flächenausnutzung [%]	3.62e-5	Substrat

Abbildung 48: *Berechnung des Merkmalssatzes*

Im vorliegenden Fall kommen bei der Merkmalsgenerierung die in Abb.32 vorgestellte Algorithmen zur Flächenberechnung zur Anwendung.

Der Merkmalsgenerator berechnet und erfaßt hierbei die Fläche jeder Einzelstruktur zusammen mit ihrem Namen und addiert den erhaltenen Wert zur Gesamtfläche. Zur Berechnung der Flächenausnutzung setzt man diese ins Verhältnis zur gegebenen Substratfläche. Bei diesem Anwendungsbeispiel wurde eine Substratfläche von 23 x 62 mm^2 vorausgesetzt.

Zur Lokalisation eines Merkmals wird jeweils der Name der betroffenen Figur herangezogen. Im Falle substratbezogener Merkmale wie der Gesamtfläche oder der Flächenbelegung, wird hierfür das Schlüsselwort „Substrat" verwendet.

Einen Auszug aus dem zu diesem Anwendungsbeispiel berechneten Merkmalssatz zeigt Abb.48.

6.3.3 Auswahl eines Regelsatzes

Die Auswahl eines Regelsatzes erfolgt über das mit „Einzulesender Regelsatz" gekennzeichnete Texteingabefeld des Diagnosekontrollfensters. Alternativ kann hierfür auch die Ladeoption des zum Schalter „Regelsatz" gehörenden Menü verwendet werden. Neben dieser Option beinhaltet das Menü noch Funktionen zur Erstellung, bzw. Bearbeitung von Regelsätzen.

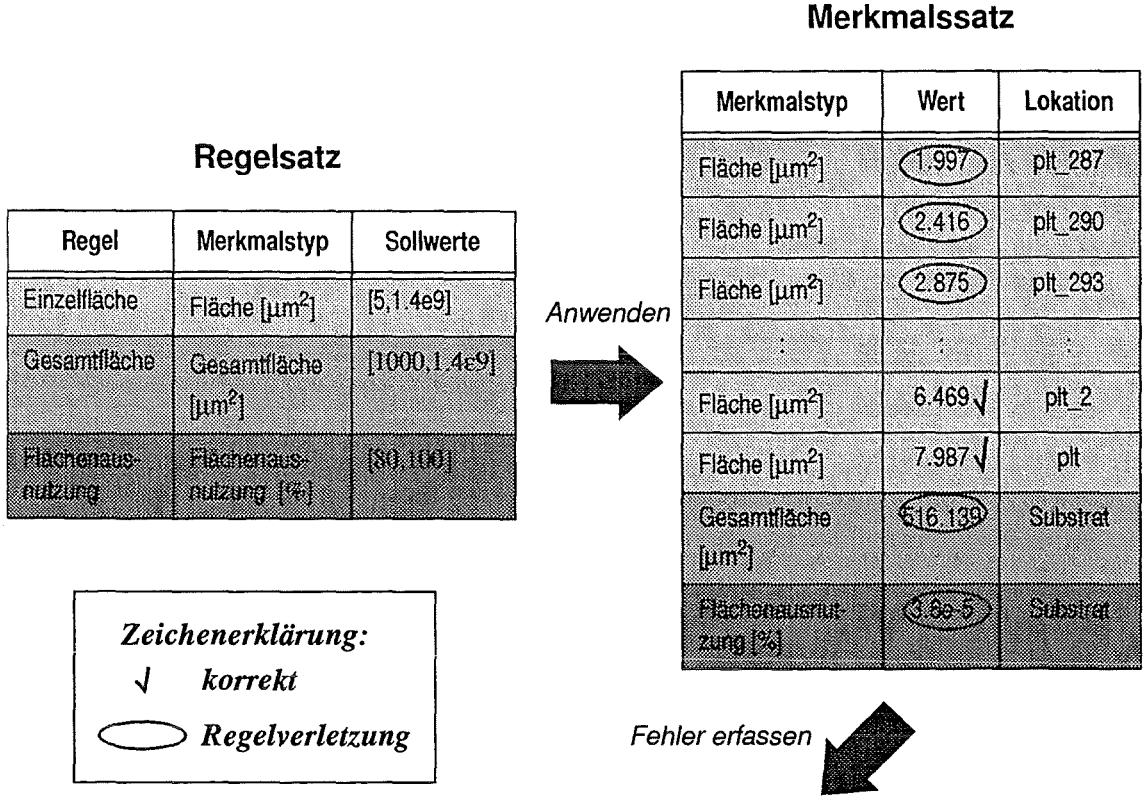
6.3.4 Diagnose

Nach der Vorgabe eines Regelsatzes und der Generierung eines Merkmalssatzes verfügt das Diagnosesystem über alle für die Durchführung einer Entwurfsregelprüfung benötigten Regeln und Fakten. Die für die Diagnose verantwortliche Inferenzmaschine kann dann über den Schalter „Entwurfsdiagnose" der in Abb.46 dargestellten Benutzeroberfläche aktiviert werden.

Den Ablauf der am vorliegenden Testentwurf vorgenommen Entwurfsregelprüfung veranschaulicht Abb.49. Das Inferenzmodul wendet die im vorgegebenen Regelsatz enthaltenen Entwurfsregeln auf die entsprechenden Merkmale an. In der Abbildung sind korrespondierende Regeln und Merkmale jeweils durch die gleiche Graustufe gekennzeichnet.

Bei der Prüfung werden die in der Merkmalsliste enthaltenen Ist-Werte mit den in den Regeln vorgegebenen Sollwerten verglichen. Im Falle einer Regelverletzung werden die Kenndaten des Merkmals und die von diesem Merkmal verletzten Regeln im Fehlerprotokoll erfaßt und in einer Datei gesichert.

Der Name der Sicherungsdatei erscheint nach Beendigung der Diagnose im Texteingabefeld „Ergebnissatz". Über dieses Eingabefeld können auch bereits früher erstellte Fehlerprotokolle zur Visualisierung geladen werden.



Regel	Merkmalstyp	Sollwerte
Einzelfläche	Fläche [µm²]	[5,1.4e9]
Gesamtfläche	Gesamtfläche [µm²]	[1000,1.4e9]
Flächenausnutzung	Flächenausnutzung [%]	[80,110]

Merkmalstyp	Wert	Lokation
Fläche [µm²]	1.997	plt_287
Fläche [µm²]	2.416	plt_290
Fläche [µm²]	2.875	plt_293
:	:	:
Fläche [µm²]	6.469 √	plt_2
Fläche [µm²]	7.987 √	plt
Gesamtfläche [µm²]	516.139	Substrat
Flächenausnutzung [%]	3.6e-5	Substrat

Fehlerprotokoll

Merkmalstyp	Wert	Lokation	Regel
Fläche [µm²]	1.997	plt_287	Einzelfläche
Fläche [µm²]	2.416	plt_296	Einzelfläche
Fläche [µm²]	2.875	plt_293	Einzelfläche
:	:	:	:
Gesamtfläche [µm²]	516.14	Substrat	Gesamtfläche
Flächenausnutzung [%]]	3.6e-5	Substrat	Flächenausnutzung

Abbildung 49: *Durchführung der Entwurfsdiagnose*

6.3.5 Darstellung der Diagnoseresultate

Den Abschluß der Entwurfsregelprüfung bildet die Interpretation der Diagnoseresultate im Dialog mit dem Konstrukteur. Das hierfür entwickelte Visualisierungswerkzeug (Abb.50) wird mit Hilfe des zum Schalter „Entwurfsdiagnose" der Diagnosesteuerung gehörenden Menüs gestartet. Hierzu ist die Option „Diagnoseergebnis ansehen" zu wählen.

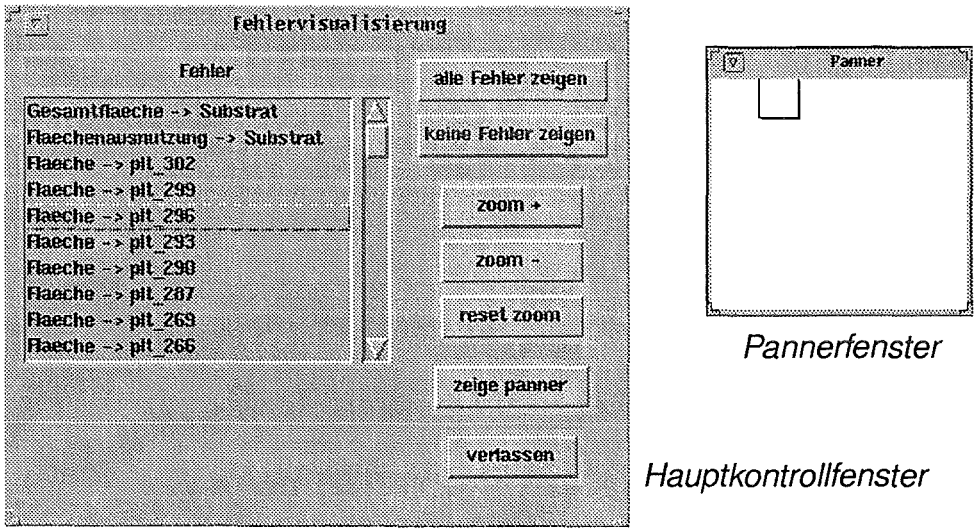


Abbildung 50: *Kontrolloberfläche der Fehlervisualisierung*

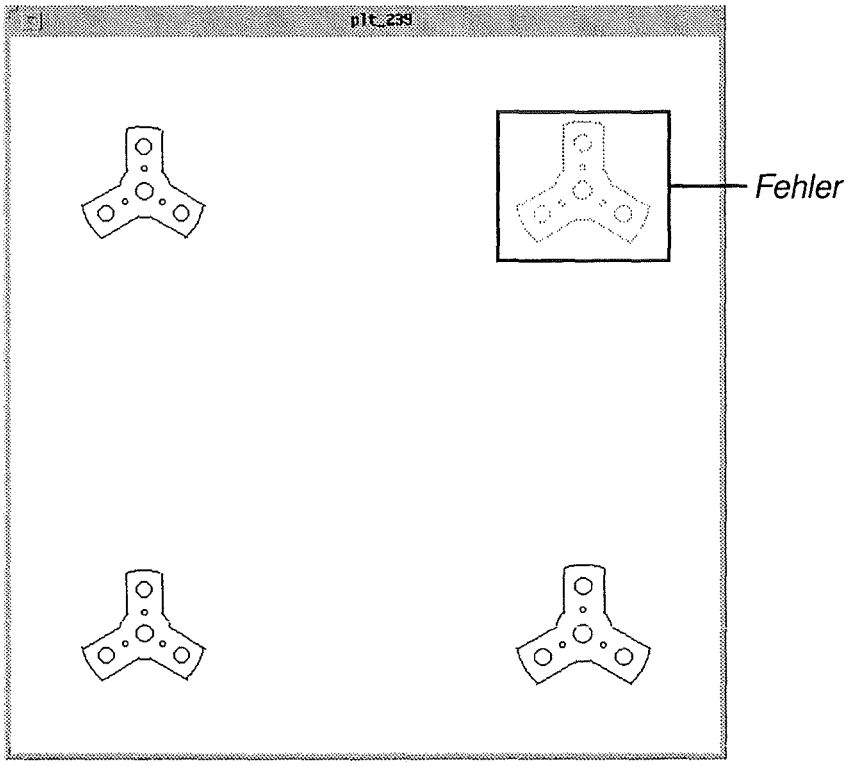
Die Oberfläche des Visualisierungswerkzeuges besteht aus einer graphischen Anzeige des bearbeiteten Entwurfes und zwei Fenstern mit Kontrollelementen zur Steuerung der Anzeige.

Die wichtigste Komponente des Hauptkontrollfenster ist eine Auswahlliste mit allen ermittelten Entwurfsfehlern. Die Listeneinträge setzen sich aus der Bezeichnung des Merkmals und dem Namen der betroffenen Figur zusammen. Bei Selektion eines Eintrages wird die dazugehörige Figur in der Entwurfszeichnung farblich hervorgehoben (siehe Abb.51a). Im Fall des in der Abbildung selektierten Eintrags handelt es sich bei dem betroffenen Merkmal um die Fläche des Planetenträgers mit der Kennzeichnung „plt_296".

Mit Hilfe der Schalter „zoom +" und „zoom -" hat der Anwender die Möglichkeit, sich den betreffenden Bildausschnitt in der gewünschten Vergrößerung anzeigen zu lassen. Zur Navigation innerhalb der Zeichnung dient der vom Pannerfenster bereitgestellte Cursor.

Als Alternative zur Anzeige einzelner Fehler kann sich der Benutzer mit dem Schalter „alle Fehler zeigen" auch sämtliche fehlerhaften Elemente seines Entwurfes gleichzeitig ansehen (siehe Abb.51b).

a) Anzeige einzelner Fehler



b) Anzeige aller Fehler

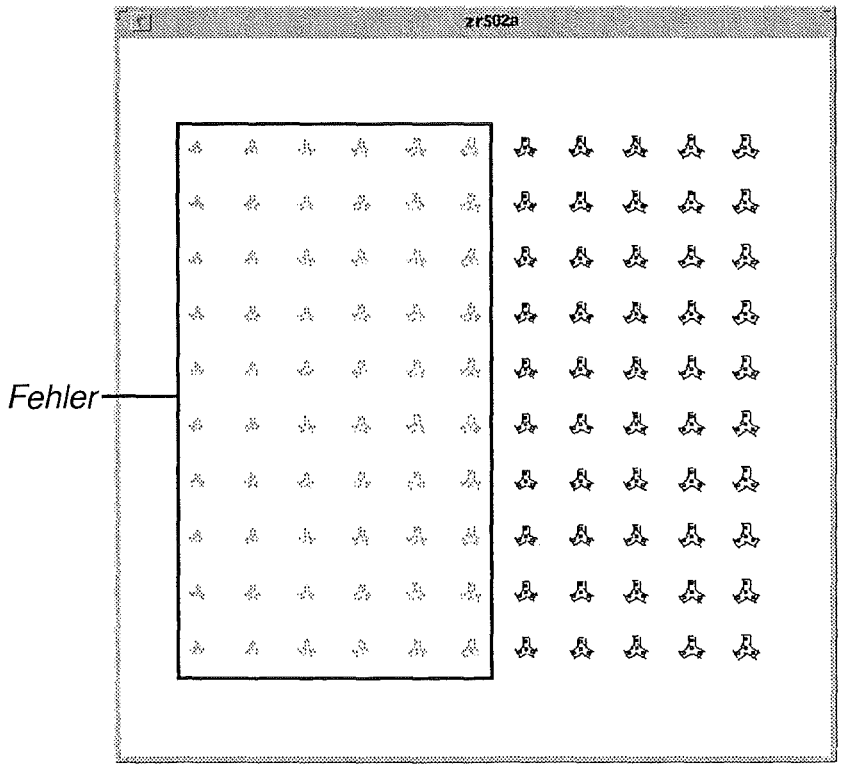


Abbildung 51: *Graphische Anzeige des Diagnoseergebnisses*

Ergänzt wird die graphische Anzeige durch ein Textfenster, das Auskunft über die Art eines Designfehlers gibt (siehe Abb.52). Dieses beinhaltet den Merkmalsnamen, eine Parameterkennzeichnung - im Falle geometrischer Merkmale entspricht diese der Lokation des Merkmals in der Entwurfszeichnung -, den Wert des Merkmals und eine Liste mit den verletzten Regeln. Die einzelnen Regeln können jeweils durch Selektion des dazugehörigen Listeneintrages angezeigt werden. Der Konstrukteur erhält so Aufschluß über den prozeßtechnischen Hintergrund der Fehlermeldung und graphische Hilfestellung zur Behebung der Fehlersituation.

Im Falle des in Abb.52 dargestellten Entwurfsfehlers würde die in Abschnitt 5.1 vorgestellte Regel „Einzelfläche" ausgegeben werden (siehe Abb.44).

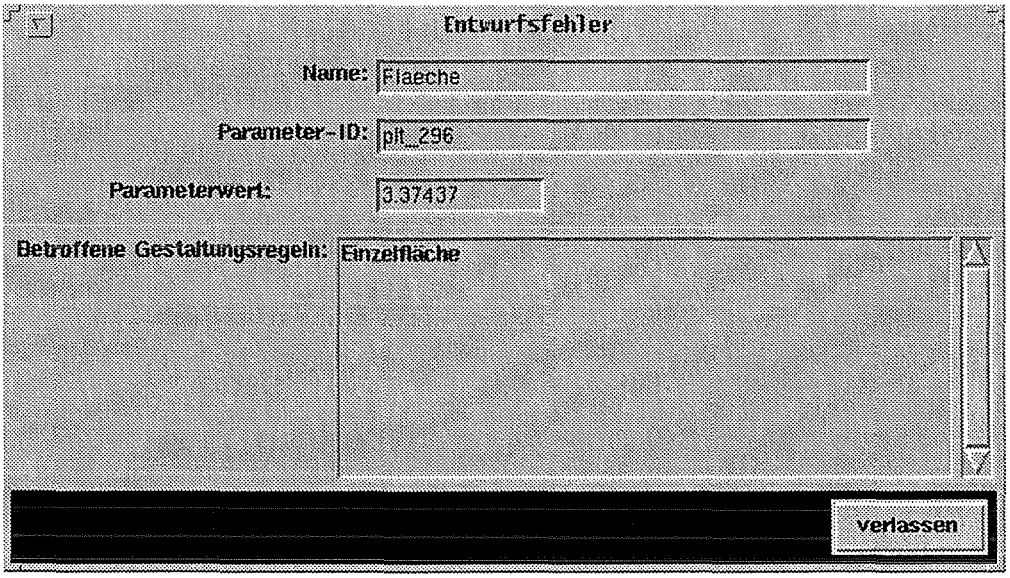


Abbildung 52: *Anzeige eines Entwurfsfehlers*

7. Zusammenfassung und Ausblick

7.1 Zusammenfassung

Das zur Mikrostrukturierung entwickelte LIGA-Verfahren befindet sich derzeit an der Schwelle zu einer industriellen Nutzung. Daraus ergeben sich in verstärktem Maße ökonomische Randbedingungen an diese Technologie. Für die Herstellung marktbeständiger Produkte ist eine optimale Nutzung der Fertigungspotentiale unabdingbar. Der Schlüssel hierzu liegt in einem fertigungsgerechten Entwurf der Strukturen, d.h. die Gestalt der Mikrostrukturen sollte - soweit von den funktionalen Anforderungen her möglich - hinsichtlich fertigungstechnischer Kriterien optimiert werden.

Um einen Entwurf im fertigungsgerechten Sinne gestalten zu können, ist der Konstrukteur auf ein entsprechendes Hintergrundwissen über die zugrundeliegende Herstellungsmethode angewiesen. Das umfangreiche zur Konstruktion von LIGA-Strukturen benötigte Fertigungswissen hat inzwischen jedoch eine Komplexität erreicht, die nicht mehr ohne eine angemessene informationstechnische Unterstützung handhabbar ist. Ein entsprechendes Werkzeug soll dem Konstrukteur mit dem im Laufe dieser Arbeit entwickelten wissensbasierten System MIDAS (Microstructure Diagnosis and Answerbook System) an die Hand gegeben werden.

Der erste Schritt zu einer auf Fertigunswissen basierenden Konstruktionsunterstützung ist, dem Konstrukteur einen für die Entwurfsvorbereitung geeigneten interaktiven Zugang zu den von ihm benötigten Informationen zu ermöglichen. Mit Hilfe eines entsprechend konzipierten *Informationssystems* ist der Konstrukteur in der Lage, sich zunächst einen allgemeinen Überblick über die Realisierbarkeit verschiedener Produktgeometrien zu verschaffen und die Einflüsse verschiedener Herstellungsvarianten auf den Entwurf abzuschätzen.

Der Charakter der von Konstrukteur benötigten Informationen hängt in starkem Maße vom gerade durchlaufenen Stadium des Konstruktionsprozesses ab. Um eine diesen Entwurfsphasen angemessene Unterstützung gewährleisten zu können, muß das Informationssystem in der Lage sein, Wissen aus semantisch völlig verschiedenartigen Bereichen bereitzustellen. Beispiele hierfür sind Gestaltungsregeln oder verschiedene Produkttypen. Die Darstellung dieser Formen von Fertigungswissen sollte sich dabei an der Begriffswelt des Konstrukteurs orientieren.

Ausgangspunkt der Entwicklung des Informationssystems war daher die Umsetzung dieses Fertigungswissens in ein auf diesen semantischen Kriterien basierendes objektorientiertes Modell. Die verschiedenen Formen von Wissen wurden darin in eine entsprechende Klassenstruktur umgesetzt.

Das Informationssystem orientiert sich in Aufbau und Funktion an dieser Struktur. Der Anwender erhält über einen darauf abgestimmten Navigationsmechanismus die Möglichkeit, gezielt auf einzelne Klassen oder Objekte zuzugreifen.

Für die Wissensakquisition wurde dieses System um zusätzliche Steuerungsele-

mente erweitert, die dem Anwender sowohl auf Klassen- als auch auf Objektebene Änderungen der Wissensbasis ermöglichen.

Einen Schritt weiter geht das Konzept einer *Entwurfsdiagnose,* wie sie bereits in der Mikroelektronik als Standardvalidierungsschritt üblich ist. Ziel einer Entwurfsdiagnose im Sinne eines fertigungsgerechten Entwurfes ist es, die Einhaltung fertigungstechnischer Randbedingungen hinsichtlich Form und Abmessungen der herzustellenden Mikrostrukturen zu gewährleisten.

Die Architektur des Diagnosesystems orientiert sich am Ablauf einer derartigen Entwurfsdiagnose, die sich aus folgenden Schritten zusammensetzt:

- Auswahl der zu beachtenden Entwurfsregeln (Regelselektion) aus der gegebenen Regelmenge
- Extraktion relevanter Geometrieparameter aus der Entwurfszeichnung (Merkmalsgenerierung)
- Vergleich von Soll- und Ist-Werten und Erstellen eines Fehlerprotokolles (Inferenz)
- Visualisierung des Fehlerprotokolles in Verbindung mit Verbesserungsvorschlägen (Fehlervisualisierung)

Das Diagnosesystem ist diesen Teilaufgaben entsprechend in unabhängige Module untergliedert, die über wohldefinierte Austauschschnittstellen miteinander kooperieren. Der Vorteil dieser Vorgehensweise liegt zum einen in der Möglichkeit einer getrennten Entwicklung und Erprobung der Teilsysteme, zum anderen in der Austauschbarkeit der Module, die eine Anpassung des Systems an neue Anforderungen ermöglicht.

Die Steuerung des kompletten Diagnoseablaufs wird über eine diesen Modulen übergeordnete Kontrolloberfläche, über die der Benutzer Zugriff auf die einzelnen Systemkomponenten erhält, abgewickelt.

Ein wesentlicher Gesichtspunkt bei der Entwicklung von MIDAS war eine offene Gestaltung der Systemarchitektur und der Wissensbasis, deren Inhalte vom Anwender situationsabhängig vorgegeben werden können. Hierdurch ist es möglich, Informations- und Diagnosekomponente auch in anderen Anwendungsfeldern einzusetzen.

7.2 Ausblick

In der vorliegenden Arbeit wurde das Konzept einer wissensbasierte Konstruktionsunterstützung entwickelt und in Form eines ersten Prototypen realisiert. Die Funktionsfähigkeit des Systems wurde dabei an Beispielen aus der Praxis demonstriert. Der Schwerpunkt lag bei dieser Entwicklung auf der Bereitstellung aller wesentlichen Systemfunktionen und der Schaffung einer flexiblen und hinsichtlich Erweiterungen offenen Systemstruktur.

Ein praktischer Einsatz des Systems erfordert die Berücksichtigung weiterer Kriterien. Bei der Gestaltung der Benutzeroberfläche wurde vorwiegend auf einfache Handhabbarkeit und die Absicherung aller Eingabefunktionen gegen Fehleingaben des Benutzer Wert gelegt. Hier sind sicherlich einige Erweiterungen der Funktionalität und eine an Beispielen aus dem kommerziellen Bereich orientierte Gestaltung der Oberfläche denkbar, die nicht im Rahmen dieser prototypischen Implementierung vorgesehen war.

Da der praktische Einsatz des Systems in der Regel im Mehrbenutzerbetrieb erfolgen wird, ist außerdem die Bereitstellung zusätzlicher Verwaltungsmechanismen, die den Zugriff auf die Wissensbasis enthaltenen Klassen und Objekte regeln und die Konsistenz der Datenbasis gewährleisten, notwendig. Daher ist es naheliegend, das System auf der Basis eines objektorientierten Datenbank Management Systems (OODBMS) zu reimplementieren. Da kommerzielle OODBM-Systeme in der Regel mit einer C++ Schnittstelle ausgestattet sind, ist eine direkte Portierung der meisten für MIDAS entwickelten Klassen möglich. Größere Änderungen sind dabei lediglich im Falle der Persistenzmechanismen zu erwarten, deren Funktionen das Datenbanksystem wahrnehmen würde.

Für diesen Schritt spricht auch ein ganz anderer Gesichtspunkt. Wie das Beispiel der Mikroelektronik zeigt, ist die erfolgreiche Umsetzung einer Technologie nur durch eine konsequente informationstechnische Unterstützung des gesamten Verfahrensablaufs möglich. Im Umfeld der LIGA-Technologie wurden mittlerweile Werkzeuge für die verschiedensten Anwendungsbereiche entwickelt oder befinden sich im Entstehen. Das Spektrum reicht dabei von der Konstruktionsunterstützung [Hahn95, Egg95] bis hin zu einer optischen Qualitätskontrolle von Mikrostrukturen [BGH91, GDGHS93]. Um das Ziel einer durchgängigen Werkzeugunterstützung des gesamten LIGA-Verfahrens zu verwirklichen ist für die Zukunft eine Integration dieser Werkzeuge in einer gemeinsamen Umgebung mit einem objektorientierten Datenbanksystem als Integrationsplattform [Wie95] erstrebenswert.

Anhang

A Implementierung von MIDAS

Die Implementierung von MIDAS erfolgte auf einer Sun SPARCstation unter Solaris 2.x [Sun93]. Als Implementierungssprache wurde dabei C++ [Str92] verwendet. Die Vorteile dieser Sprache liegen zum einen in der Unterstützung objektorientierter Konzepte, zum anderen in der hohen Kompatibilität zu C [KR90] das im UNIX-Umfeld zur Systemprogrammierung eingesetzt wird [Gul88,Ill92] und seiner großen Verbreitung entsprechend in einem weiten Spektrum an Werkzeugen und Bibliotheken Unterstützung findet.

Zur Realisierung grundlegender Datenstrukturen wie Listen oder Dictionaries wurde auf die Klassenbibliothek Tools.h++ Version 6 von Rogue Wave Software [RWS94] zurückgegriffen, die auch für andere Plattformen wie z.B. verschiedene PC-Betriebsysteme erhältlich ist.

Die Benutzeroberfläche des Systems basiert auf der Philosophie, daß jedes darzustellende Objekt selbst für seine Visualisierung verantwortlich ist. Die dafür benötigten Oberflächenelemente wurden mit Hilfe des XView-Toolkits [Hel91, VR91] erstellt. Zur Entkopplung des übrigen Programmcodes von diesen plattformabhängigen Elementen wurden Schnittstelle zur Benutzeroberfläche in einem dafür entwickelten Satz von Visualisierungsklassen gekapselt. Bei einer Portierung des Systems auf eine andere Benutzeroberfläche ist so nur eine Anpassung dieser speziellen Klassen erforderlich.

Lediglich bei der Initiierung der Oberfläche wird direkt auf Funktionen und Objekte des XView-Paketes zurückgegriffen. Hierbei kam der zu XView gehörige Code-Generator OpenWindows Developer's Guide [Sun94a,Sun94b] zum Einsatz. Dieses CASE-Tool erzeugt aus einer interaktiv erstellbaren graphischen Spezifikation der gewünschten Benutzeroberfläche den dazugehörigen C oder C++Quellcode, in den eigene Funktionen integriert werden können.

Die Persistenzmechanismen von MIDAS basieren direkt auf Funktionen des UNIX-Dateisystems. Ebenso wie bei der Benutzeroberfläche wurden dabei alle systemspezifischen Elemente in einem eigenen Satz von Klassen eingeschlossen, um die Portabilität der Software zu erhöhen.

Die Beschreibung der für MIDAS entwickelten Klassen erfolgt in den sich anschließenden Kapiteln.

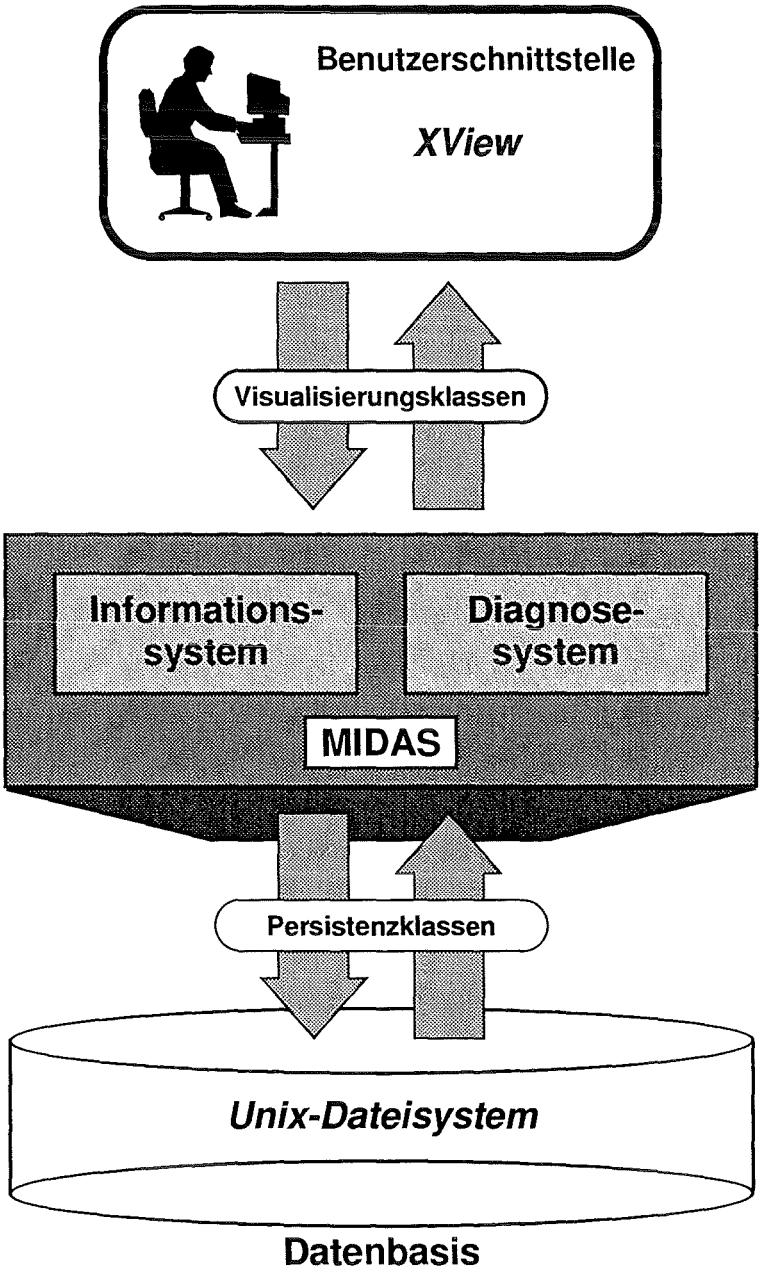


Abbildung 53: Struktur des Gesamtsystems

B Das zur Wissensrepräsentation verwendete Datenmodell

B.1 Repräsentation von Wissensklassen und ihren Instanzen

Im Falle des für den Entwurf von LIGA-Strukturen benötigten Fertigungswissens hat man es mit semantisch äußerst verschiedenartigen Wissensbereichen und dementsprechend auch mit einer sehr heterogen ausgeprägten Klassenstruktur zu tun. Um eine einheitliche datentechnische Behandlung derart unterschiedlich strukturierter Klassen von Fertigungswissen innerhalb der übergeordneten Steuerstruktur des Informationssystems zu ermöglichen, müssen diese Klassen auf einen gemeinsamen Ursprung zurückgeführt werden. Üblich ist in einem solchen Fall die in Abschnitt 4.3 vorgestellte Verfahrensweise - man führt eine abstrakte Basisklasse ein, von der die zusammenzufassenden Klassen abgeleitet werden. Hierdurch kann eine Anwendung jede Instanz dieser Klassen auch als Instanz ihrer gemeinsamen Oberklasse ansprechen.

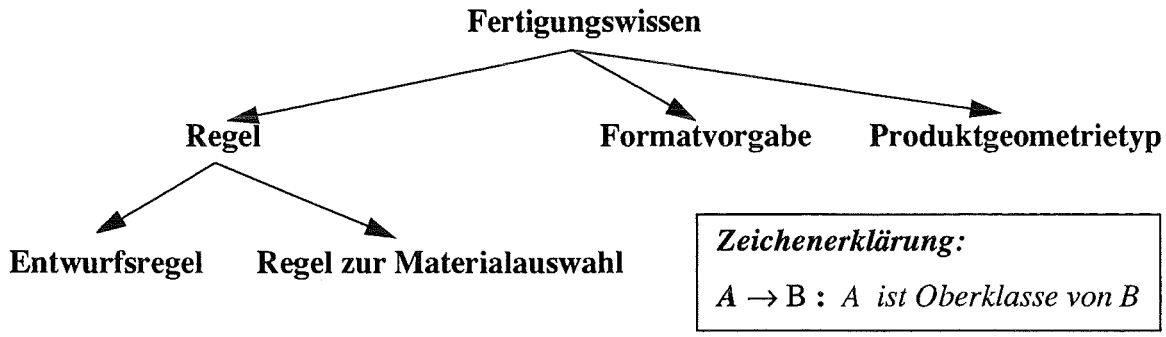


Abbildung 54: Modellierung des für den Entwurf benötigten Fertigungswissens

Aufgrund der forschreitenden Entwicklung der LIGA-Technologie unterliegt das zur Entwurfsunterstützung benötigte Fertigungswisssen jedoch häufigen Änderungen, die sich nicht nur auf die Wissensinhalte selbst sondern auch auf die zugrundeliegende Klassenstruktur auswirken können. Gegebenenfalls müssen neue Klassen eingeführt oder bereits bestehende Klassen modifiziert werden. Im Falle der oben geschilderten Vorgehensweise hätte dies die Konsequenz, daß bei jeder Änderung der Klassenstruktur eine entsprechende Anpassung der Implementierung erforderlich wäre. Sinnvoller ist es natürlich, dem Anwender selbst die Aktualisierung seiner Wissensbasis zu ermöglichen, wofür er allerdings zur Programmlaufzeit Zugriff auf das einer Wissensklasse zugrundeliegende Attributschema benötigt, was bei C++ nicht möglich ist.

Aus diesem Grund wird bei MIDAS zur Wissenmodellierung eine Klasse „KnowledgeObject" eingeführt, deren Instanzen ihre Informationen in einer variablen Attributli-

ste verwalten. Anzahl und Typ der Attribute können so den jeweils zu repräsentierenden Informationen angepaßt werden.

Zur Darstellung von Attributen verschiedener Datentypen wie Text, Grafik oder numerischer Werte, dient ein entsprechender Satz von Attributklassen, die von einer Basisklasse „Attribute" abgeleitet sind.

Das der Liste zugrundeliegende Attributschema wird über eine Instanz der hierfür konzipierten Klasse „KnowledgeObjectScheme" festgelegt. Einer Klasse-Objektbeziehung auf der Ebene des Fertigungswissens entspricht in diesem Modell die Zuordnung eines Attributschemas zu einem Objekt.

Als Träger der Schemainformation fungiert eine Liste von Attributkonfigurationen, die von Instanzen der Klasse „AttributeSetup", bzw. ihrer Unterklassen repräsentiert werden. Darin sind der Attributtyp, sowie der zulässige Wertebereich der dazugehörigen Attribute festgelegt (siehe Abb.55).

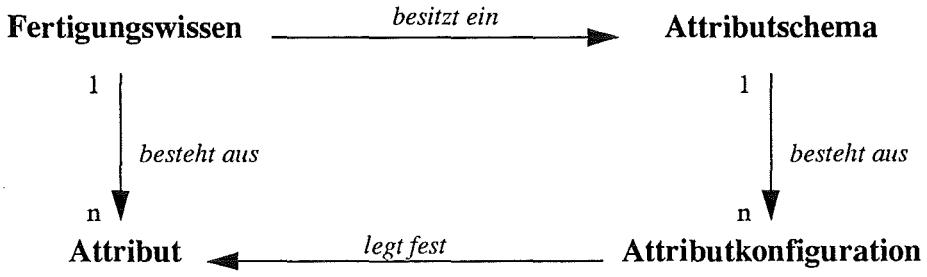


Abbildung 55: *Wissensrepräsentation bei MIDAS*

Die zur Umsetzung dieses Modells entwickelte Klassenstruktur zeigt Abb.56 in der in [CY91] verwendeten Notation.

B.2 Klassen zur Wissensrepräsentation

B.2.1 Die Klasse KnowledgeObject

Die Klasse KnowledgeObject dient der Verwaltung der zu einem Wissensbereich gehörenden Informationen mit Hilfe einer variablen Attributliste. Anzahl und Typ der Attribute werden durch ein über den Zeiger auf eine Instanz der Klasse KnowledgeObjectScheme vorgegebenes Atttributschema festgelegt.

Enthält ein Wissensbereich verifizierbares Regelwissen, besteht die Möglichkeit, in einer Liste Wertebereiche für die davon betroffenen Entwurfsparameter vorzugeben und deren Werte im Rahmen einer Entwurfsdiagnose zu überprüfen.

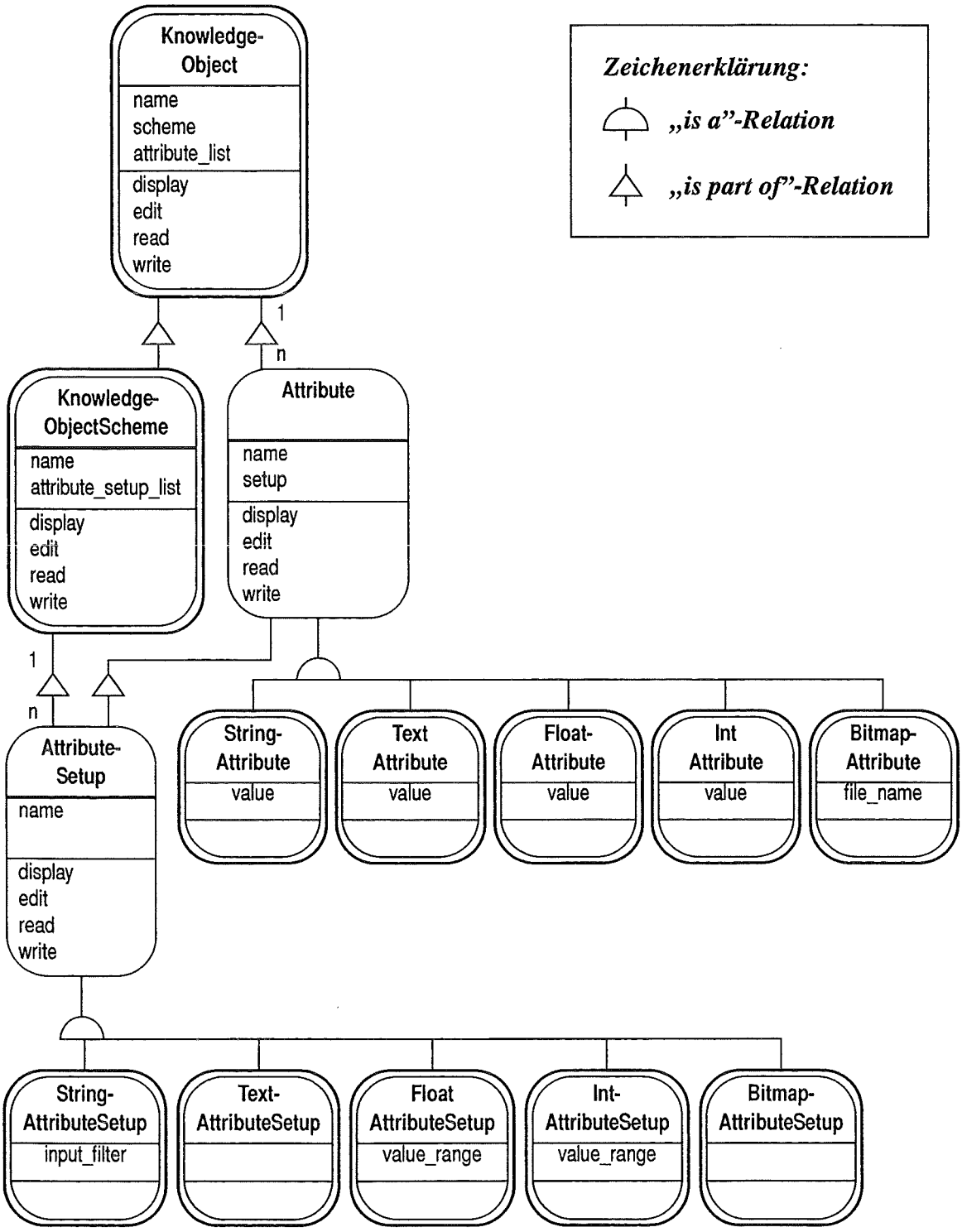


Abbildung 56: Klassen zur Wissensrepräsentation

B.2.2 Die Klasse KnowledgeObjectScheme

Mithilfe von Instanzen dieser Klasse wird das Attributschema eines Wissensbereiches, also die Zusammensetzung der Attributliste einer Instanz der Klasse KnowledgeObject definiert. Diese Schemainformation besteht aus einer Liste von Attributesetups, in denen Typ und erlaubter Wertebereich eines Attributes definiert sind. Gleichzeitig stellen diese Attributesetups entsprechend konfigurierte Bildschirmelemente zur Anzeige, bzw. Eingabe der Attributwerte und zur Verfügung, auf deren Basis die Klasse KnowledgeObjectScheme ein Bildschirmformular zur Anzeige oder Bearbeitung von Wissensbereichen dieses Schemas generieren kann.

Bei Schemaänderungen werden die zugeordneten Wissensbereiche an das neue Schema angepaßt.

B.2.3 Die Klasse Attribute

Zur Darstellung von Attributen verschiedener Datentypen wie Text, Grafik oder numerischer Werte dient den Instanzen der Klasse KnowledgeObject ein entsprechender Satz von Attributklassen. Bei der Klasse Attribute handelt es sich um deren Basisklasse, in der die Methoden zur Bildschirmeingabe bzw. -anzeige und für den Zugriff auf die I/O-Streams definiert sind.

B.2.4 Die Klasse StringAttribute

Mithilfe der Klasse StringAttribute können Zeichenketten über die Benutzeroberfläche oder die I/O-Streams ein-, bzw. ausgegeben werden. Die hierbei zulässigen Zeichen und die maximale Länge der Zeichenkette werden über eine zugeordnete Instanz der Klasse StringAttributeSetup festgelegt, welche auch die benötigten Elemente der Benutzeroberfläche bereitstellt.

B.2.5 Die Klasse TextAttribute

Aufgabe der Klasse TextAttribute ist das Einlesen, bzw. die Ausgabe mehrzeiliger, aus beliebigen Zeichen zusammengesetzter Texte über die Benutzeroberfläche oder die I/O-Streams. Die maximale Länge des Textes wird über eine zugeordnete Instanz der Klasse TextAttributeSetup festgelegt, welche auch die benötigten Elemente der Benutzeroberfläche bereitstellt. Im Gegensatz zur Klasse StringAttribute werden die vom Benutzer eingegebenen Zeichen jedoch bei dieser Klasse keiner weiteren Kontrolle unterzogen.

B.2.6 Die Klasse FloatAttribute

Mithilfe der Attribute-Unterklasse FloatAttribute können Fließkommazahlen über die Benutzeroberfläche oder die I/O-Streams ein-, bzw. ausgegeben werden. Der zulässige Wertebereich eines Attributes wird über eine zugeordnete Instanz der Klasse FloatAttributeSetup festgelegt, welche auch die benötigten Elemente der Benutzeroberfläche bereitstellt.

B.2.7 Die Klasse IntAttribute

Bei der Klasse IntAttribute handelt es sich um eine Unterklasse von Attribute, die zur Ein-, bzw. Ausgabe von ganzen Zahlen über I/O-Streams und Benutzeroberfläche dient. Die Festlegung des Wertebereichs eines ganzzahligen Attributes und die Bereitstellung der benötigten Oberflächenbausteine erfolgen mit Hilfe einer Instanz der Klasse IntAttributeSetup.

B.2.8 Die Klasse BitmapAttribute

Mit Hilfe der Klasse BitmapAttribute können Bilddateien im PBM-Format über die Benutzeroberfläche geladen und angezeigt werden. Die Größe der Bildausgabefläche wird über eine zugeordnete Instanz der Klasse BitmapAttributeSetup festgelegt, welche auch die benötigten Elemente der Benutzeroberfläche bereitstellt.

B.2.9 Die Klasse AttributeSetup

Um unzulässige Wertzuweisungen bei der Eingabe von Attributwerten auszuschließen, wird jeder Instanz einer Attributklasse ein Wertebereich zugewiesen. Das hierfür benötigte Instrumentarium wird in Form der Klasse AttributeSetup und ihrer Unterklassen bereitgestellt.

B.2.10 Die Klasse StringAttributeSetup

Die Aufgabe der Klasse StringAttributeSetup besteht darin, unzulässige Wertzuweisungen bei den Eingabeoperationen der Klasse StringAttribute auszuschließen. Hierfür stellt die Klasse StringAttribueSetup eine entsprechend konfigurierte Benutzeroberfläche und Methoden für den I/O-Stream-Zugriff zur Verfügung, die nicht zugelassene Zeichen unterdrücken und die Länge der eingegebenen Zeichenkette auf einen vorgegebenen Wert beschränken.

B.2.11 Die Klasse TextAttributeSetup

Die Klasse TextAttributeSetup dient dazu, die Rahmenbedingungen für die Eingabeoperationen der Klasse TextAttribute festzulegen. Die Klasse TextAttribueSetup stellt hierfür eine zur Eingabe mehrzeiliger Texte konfigurierbare Benutzeroberfläche und Methoden für den I/O-Stream-Zugriff bereit.

B.2.12 Die Klasse FloatAttributeSetup

Aufgabe der Klasse FloatAttributeSetup ist es, unzulässige Wertzuweisungen bei den Eingabeoperationen der Klasse FloatAttribute auszuschließen. Dafür stellt die Klasse FloatAttribueSetup eine entsprechend konfigurierte Benutzeroberfläche und Methoden für den I/O-Stream-Zugriff zur Verfügung, die Eingabewerte, die außerhalb eines vorgegebenen Wertebereichs liegen, unterdrücken.

B.2.13 Die Klasse IntAttributeSetup

Die Klasse IntAttributeSetup dient der Kontrolle aller über Benutzeroberfläche oder Streams erfolgenden Eingabeoperationen der Klasse IntAttribute. Eingabewerte, die außerhalb eines vorgegebenen Wertebereichs liegen, werden von den dafür bereitgestellten Prüfmethoden unterdrückt.

B.2.14 Die Klasse BitmapAttributeSetup

Die Aufgabe der Klasse BitmapAttributeSetup besteht darin, die Rahmenbedingungen für die Eingabeoperationen der Klasse BitmapAttribute festzulegen. Hierfür stellt die Klasse BitmapAttribueSetup eine zum Laden und Anzeigen von Bilddateien im PBM-Format konfigurierbare Benutzeroberfläche zur Verfügung.

C Zur Entwurfsdiagnose verwendete Klassen

C.1 Erweiterungen des Datenmodells für die Entwurfsdiagnose

Entwurfsregeln werden bei MIDAS wie alle dem Fertigungswissen zugehörige Objekte mit Hilfe der für das Informationssystem entwickelten Klassen von Informationsträgern repräsentiert. Deren Verwendung im Rahmen einer Entwurfsdiagnose erfordert eine Erweiterung des dort vorgestellten Konzeptes um die Möglichkeit, eine beliebige Zahl von Entwurfsrandbedingungen mit einer Regel zu verknüpfen. Aus diesem Grund wurde die zur Wissensrepräsentation entwickelte Klasse KnowledgeObject um ein zusätzliches Attribut, eine variable Liste von Entwurfsrandbedingungen, erweitert.

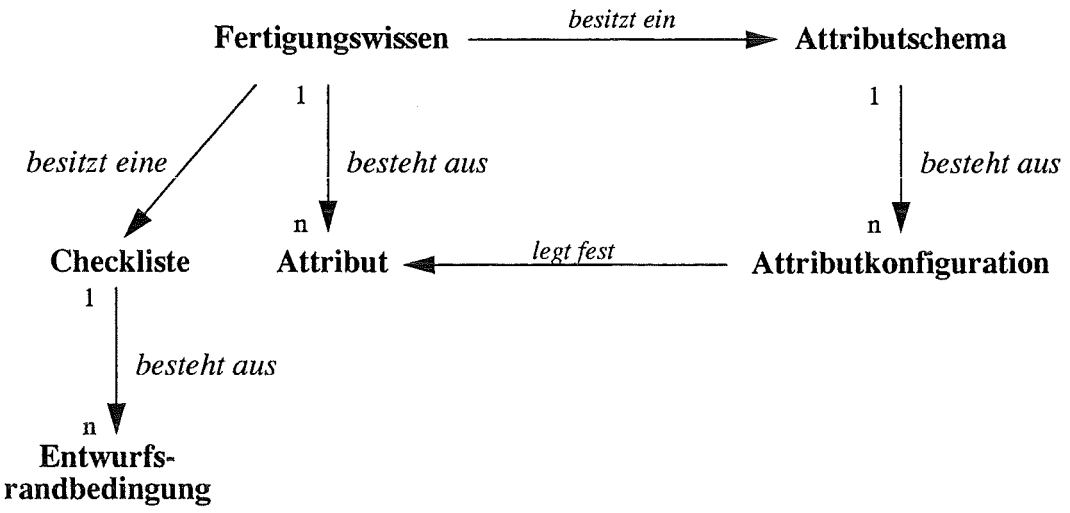


Abbildung 57: *Erweiterung des Datenmodells für die Entwurfsdiagnose*

Diese Liste verfügt über eine eigene Benutzeroberfläche, mit deren Hilfe neue Randbedingungen hinzugefügt oder vorhandene modifiziert, bzw. gelöscht werden können. Bei der Entwurfsdiagnose werden diese Randbedingungen sequentiell, also in der Art einer Checkliste verifiziert.

Die zur Umsetzung dieses Modells entwickelte Klassenstruktur zeigt Abb.56 in der in [CY91] verwendeten Notation.

C.1.1 Die Klasse ParBounds

Die Klasse ParBounds dient zur Festlegung und Verifikation der für einen Entwurfsparameter geltenden Randbedingungen. Die Form der Randbedingungen hängt vom Datentyp des betroffenen Entwurfsparameters ab. So kann bei numerischen Parametern beispielsweise ein Intervall als zulässiger Wertebereich angegeben werden,

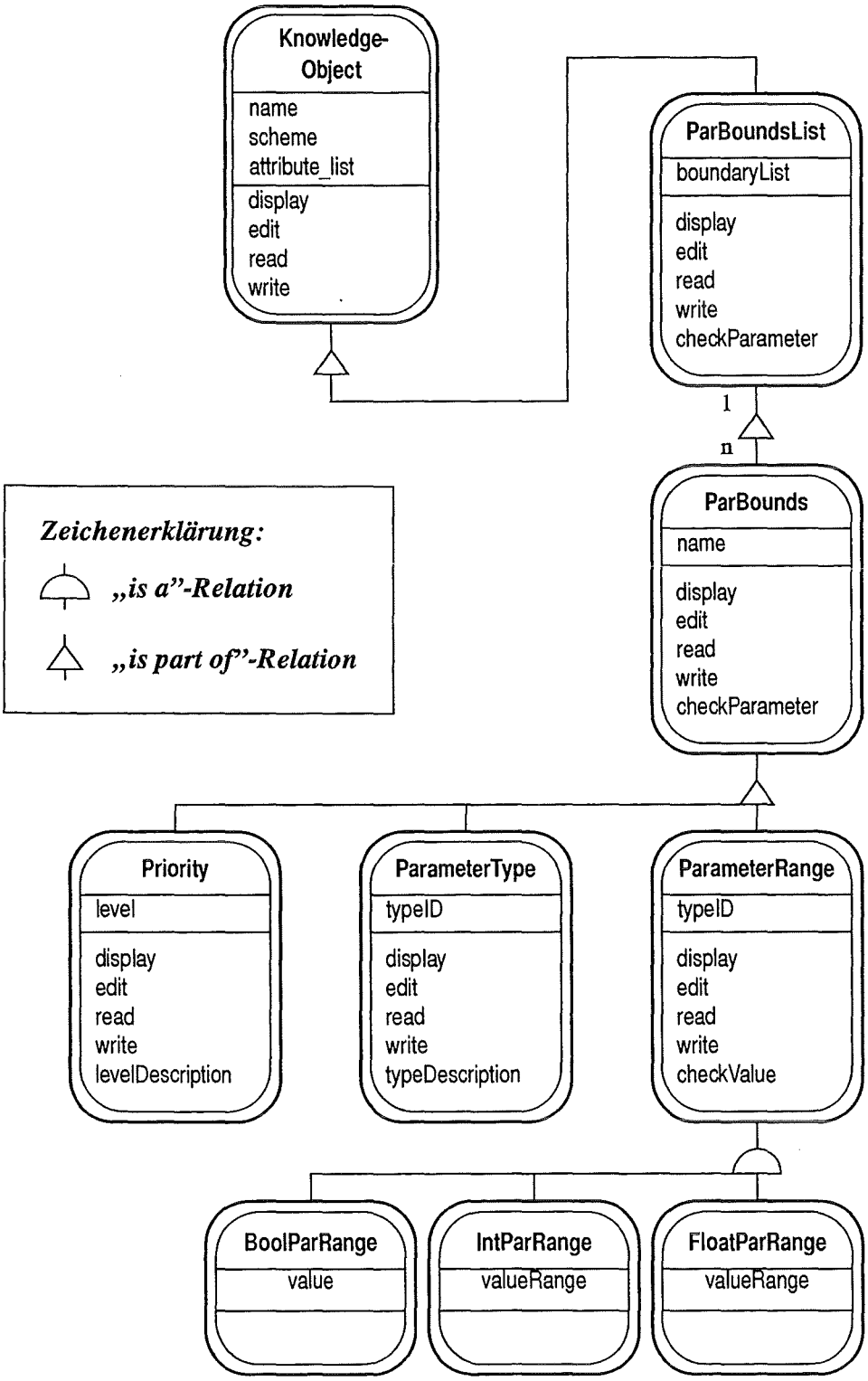


Abbildung 58: Klassen zur Modellierung von Entwurfsrandbedingungen

bei Boole'schen Parametern ist dieser Wertebereich dagegen auf einen einzigen Wert beschränkt.

Zur Auflösung möglicher Konflikte innerhalb der für die Diagnose eines Entwurfs herangezogenen Regelmenge bietet die Klasse ParBounds die Möglichkeit, die Priorität einer Randbedingung festzulegen.

Name, Typ und zulässiger Wertebereich des betroffenen Entwurfsparameters, sowie die Priorität der Randbedingung können über die Benutzeroberfläche der Klasse ausgegeben, bzw. geändert werden. Daneben verfügt die Klasse über Methoden für den Zugriff auf die I/O-Streams.

C.1.2 Die Klasse ParBoundsList

Aufgabe der Klasse ParBoundsList ist die Verwaltung der einer Entwurfsregel zugeordneten Entwurfsrandbedingungen - also Objekten der Klasse ParBounds - in Form einer einfach verketteten Liste. Hierfür stellt die Klasse eine Oberfläche zur Anzeige, bzw. interaktiven Bearbeitung der Liste und Methoden zur Sicherung des Listeninhaltes zur Verfügung.

C.1.3 Die Klasse Priority

Die Klasse Priority dient zur Festlegung der Prioritätsstufe einer Randbedingung für den Entwurf einer Mikrostruktur. Die Prioritätsstufe kann über die Benutzeroberfläche angezeigt, bzw. geändert werden. Daneben stellt die Klasse Methoden für den Zugriff auf die I/O-Streams zur Verfügung.

C.1.4 Die Klasse ParameterType

Mit Hilfe der Klasse ParameterType wird der Datentyp eines Entwurfparameters festgelegt. Der Datentyp kann über die Benutzeroberfläche angezeigt, bzw. geändert werden. Daneben stellt die Klasse Methoden für den Zugriff auf die I/O-Streams zur Verfügung.

C.1.5 Die Klasse ParameterRange

Die Klasse ParameterRange ist die Basisklasse eines Satzes von Klassen, die zur Festlegung des Wertebereiches eines Entwurfsparameters dienen. Die Basisklasse definiert den typunabhängigen Teil der Methodenschnittstelle.

C.1.6 Die Klasse BoolRange

Die Klasse BoolRange dient der Festlegung des Sollwerts eines Entwurfsparameters vom Datentyp Bool und stellt die zur Anzeige, bzw. Änderung des Wertes benötigte Benutzeroberfläche zu Verfügung.

C.1.7 Die Klasse IntRange

Mit Hilfe der Klasse IntRange wird der zulässige Wertebereichs eines ganzzahligen Entwurfsparameters festgelegt. Die Klasse stellt außerdem auch die zur Anzeige, bzw. Änderung des Wertebereichs benötigte Benutzeroberfläche zu Verfügung.

C.1.8 Die Klasse FloatRange

Die Klasse FloatRange dient der Festlegung des zulässigen Wertebereichs eines reellwertigen Entwurfsparameters und stellt die zur Anzeige, bzw. Änderung dieses Wertebereichs benötigte Benutzeroberfläche zu Verfügung.

C.2 Klassen zur Modellierung von Merkmalen

Zur Durchführung einer Entwurfsdiagnose werden abprüfbare Größen benötigt, die vollautomatisch gegen die in einer Regel enthaltenen Referenzgrößen verglichen werden können. Bei MIDAS erfolgt die Repräsentation des zu prüfenden Faktenwissens durch Instanzen verschiedener Merkmalsklassen (siehe Abschnitt 5.3.2).

Die Umsetzung des zur Repräsentation von Merkmalen entwickelten Modells in eine entsprechende Klassenstruktur zeigt Abb.59 in der in [CY91] üblichen Notation.

C.2.1 Die Klasse DesignParameter

Die Klasse DesignParameter ist Basisklasse der verschiedenen zur Darstellung von geometrie-, prozeß- oder materialbezogenen Merkmalen vorgesehenen Klassen. In der Basisklasse sind alle vom Merkmalstyp unabhängigen Elemente zusammengefaßt.

Attribute der Klasse DesignParameter sind der Merkmalsname, eine Kennzahl für den Merkmalstyp, der Merkmalswert und ein Verweis auf ein mittels einer ParameterID gekenzeichnetes Bezugsobjekt. Beispielsweise im Falle eines geometrischen Merkmals entspricht dieser Identifikator der Lokation des Merkmals innerhalb der Entwurfsgeometrie.

Die Methodenschnittstelle der Klasse enthält Funktionen für den Zugriff auf die Benutzeroberfläche und die I/O-Streams.

C.2.2 Die Klasse GeoDesignParameter

Aufgabe der Klasse GeoDesignParameter ist die Darstellung geometriebezogener Merkmale. Die Lokation eines Merkmals wird dabei mit Hilfe einer Instanz der Klasse GeoParameterID festgelegt.

C.2.3 Die Klasse ParameterID

Mit Hilfe der von ParameterID abgeleiteten Unterklassen wird das Bezugsobjekt eines Merkmals spezifiziert. Die Art des Bezugsobjektes hängt vom semantischen

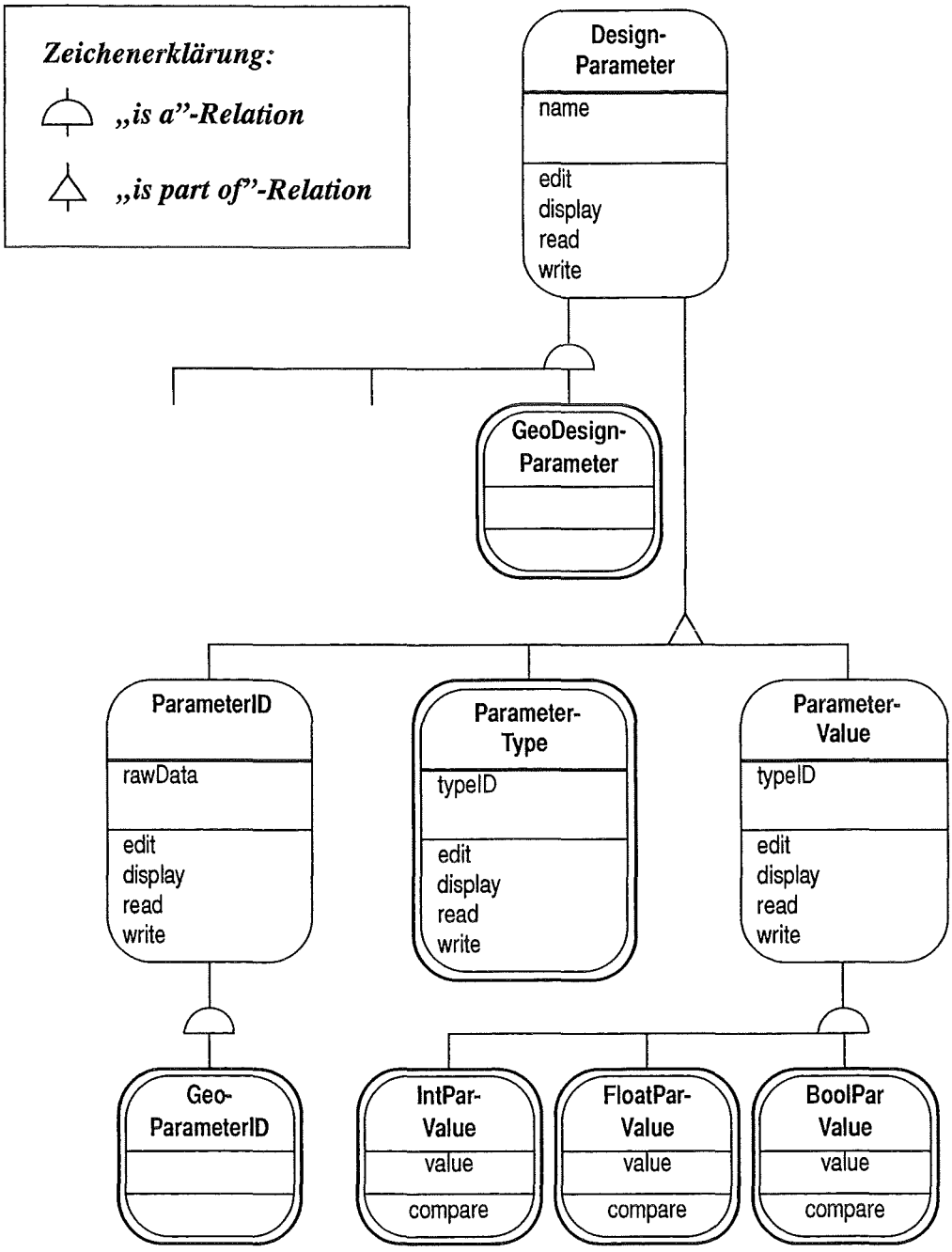


Abbildung 59: *Klassen zur Modellierung von Merkmalen*

Kontext der jeweiligen Merkmalsklasse ab.

C.2.4 Die Klasse GeoParameterID

Die Klasse GeoParameterID dient der Beschreibung des geometrischen Bezugsobjektes eines Merkmals, also seiner Lokation innerhalb der Entwurfsgeometrie. Die Beschreibung der Lokation erfolgt in Form einer Zeichenkette, die den Namen der betroffenen Figur(en) enthält. Daneben stellt die Klasse Methoden für den Zugriff auf die I/O-Streams und zur Anzeige bzw. Bearbeitung des Identifikatorwertes mit Hilfe der Benutzeroberfläche zur Verfügung.

C.2.5 Die Klasse ParameterType

Mit Hilfe der Klasse ParameterType wird der Datentyp eines Entwurfparameters festgelegt. Der Datentyp kann über die Benutzeroberfläche angezeigt, bzw. geändert werden. Daneben stellt die Klasse Methoden für den Zugriff auf die I/O-Streams zur Verfügung.

C.2.6 Die Klasse ParameterValue

Die Klasse ParameterRange ist die Basisklasse eines Satzes von Klassen, die zur Festlegung des Wertes eines Entwurfsparameters dienen. Die Basisklasse definiert den typunabhängigen Teil der Methodenschnittstelle.

C.2.7 Die Klasse BoolParValue

Die Klasse BoolParValue dient der Repräsentation von Boole'schen Merkmalswerten und stellt die zur Anzeige, bzw. Änderung des Wertes benötigte Benutzeroberfläche zu Verfügung.

C.2.8 Die Klasse IntParValue

Mit Hilfe der Klasse IntParValue werden ganzzahlige Merkmalswerte dargestellt. Die Klasse beinhaltet außerdem auch die zur Anzeige, bzw. Änderung des Wertebereichs benötigte Benutzeroberfläche.

C.2.9 Die Klasse FloatParValue

Mit Hilfe der Klasse FloatParvalue werden reellzahlige Merkmalswerte beschrieben. Außerdem stellt die Klasse die zur Anzeige, bzw. Änderung des Wertes benötigte Benutzeroberfläche zu Verfügung.

C.3 Klassen zur Repräsentation von Entwurfsfehlern

Bei einer Entwurfsdiagnose werden die ermittelten Merkmalswerte mit den im Regelsatz vorgegebenen Sollwerten verglichen und dabei festgestellte Entwurfsfehler in einem Fehlerprotokoll unter Bezugnahme auf das betroffene Merkmal und die ver-

letzten Regeln erfaßt. Die Umsetzung des in Abschnitt 5.5.1 vorgestellten Modells in eine entsprechende Klassenstruktur zeigt Abb.60.

C.3.1 Die Klasse CheckResult

Mit Hilfe der Klasse CheckResult werden bei einer Entwurfsdiagnose alle zur Spezifikation eines Entwurfsfehlers benötigten Informationen erfaßt. Hierzu gehören das betroffene Merkmal und die im Zusammenhang mit diesem Merkmal verletzten Gestaltungsregeln.

Die Methodenschnittstelle der Klasse ermöglicht den Zugriff auf die I/O-Streams und die Anzeige eines Entwurfsfehlers.

C.3.2 Die Klasse CheckResultSet

Die Klasse CheckResultSet dient der Erfassung und Darstellung aller bei einer Entwurfsdiagnose ermittelten Fehler. Diese können über die Benutzeroberfläche angezeigt und einzeln abgefragt werden. Daneben verfügt die Klasse über Methoden für den I/O-Steam- bzw. Dateizugriff.

.

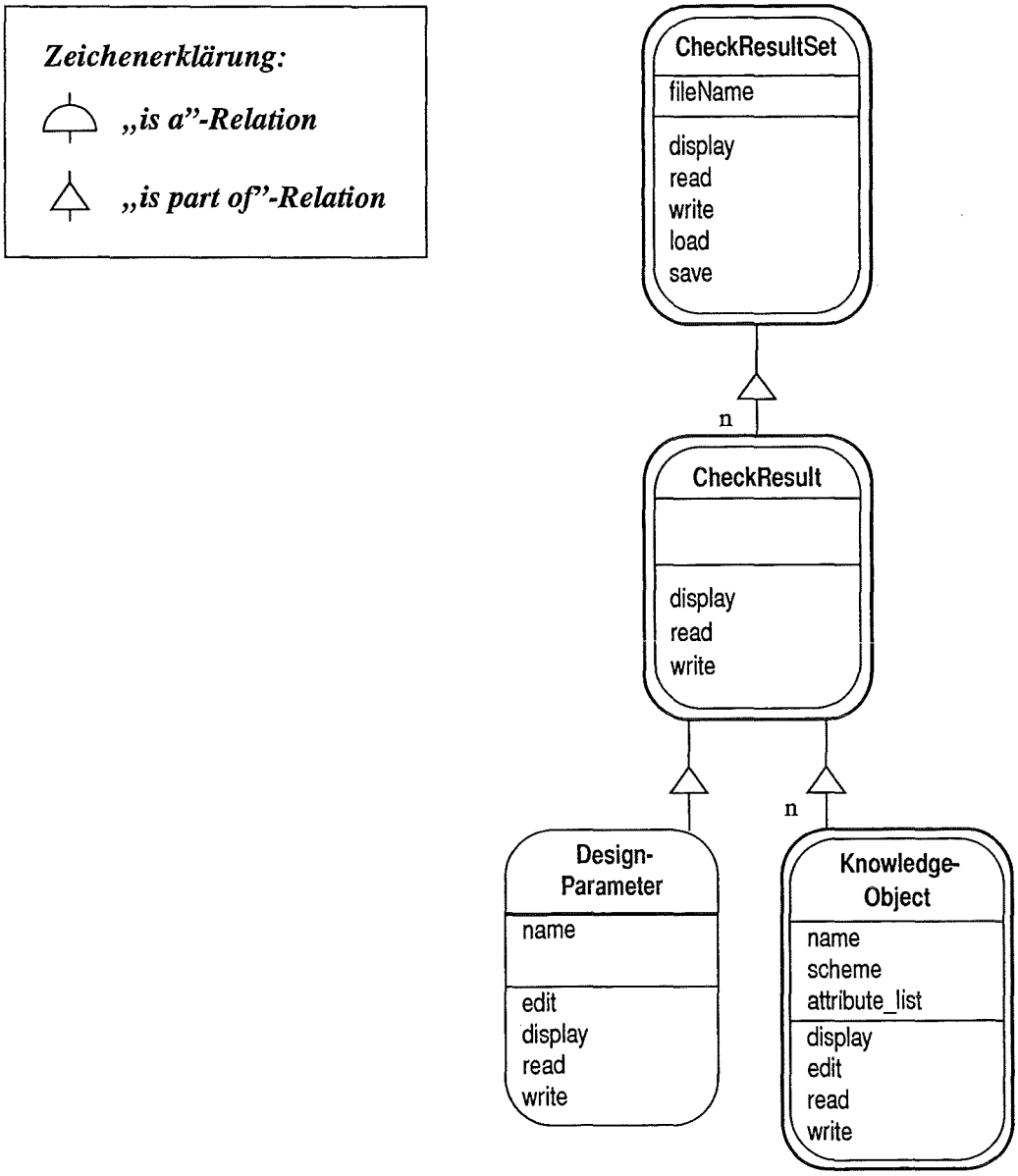


Abbildung 60: *Klassen zur Darstellung von Entwurfsfehlern*

D Die Benutzeroberfläche von MIDAS

D.1 Das Konzept der Benutzeroberfläche

Die Benutzeroberfläche von MIDAS basiert auf der Philosophie, daß jedes Objekt der Wissensbasis für seine Bildschirmdarstellung selbst verantwortlich ist. Um eine Entkopplung dieser Funktionalität von systemabhängigen Elementen zu erreichen, wurde die Schnittstelle zur verwendeten XView-Benutzeroberfläche in einen Satz spezieller Visualisierungsklassen eingebettet. Bei der Portierung von MIDAS auf eine andere Oberfläche können Änderungen so im wesentlichen auf diese Klassen begrenzt werden.

In einer etwas vereinfachten Betrachtungsweise besteht eine graphische Benutzeroberfläche aus einem oder mehreren Bildschirmfenstern und darauf plazierten Oberflächenbausteinen. Letztere dienen in der Regel der Programmsteuerung oder der Visualisierung, bzw. Bearbeitung von Informationen.

In Anlehnung an diese einfache Grundstruktur setzt sich die Oberfläche von MIDAS im wesentlichen aus Instanzen zweier entsprechender Kategorien von Visualisierungsklassen zusammen.

Auf elementarer Ebene dienen sogenannte Oberflächenbausteine der Anzeige, bzw. Bearbeitung von Informationsträgern unterschiedlichen Typs. Hierbei kann es sich um Ressourcen einfacher Datentypen wie ganze und reelle Zahlen, Boole'sche Werte, Zeichenketten, etc. handeln oder auch um komplexere Datenstrukturen wie Listen und Grafiken. Die verschiedenen, typbezogenen Klassen von Oberflächenbausteinen sind von einer abstrakten Basisklasse abgeleitet. Diese beinhaltet sämtliche zur Visualisierung eines Informationsträgers benötigten typunabhängigen Komponenten wie dessen Namen und verschiedene oberflächenspezifische Parameter. Die Methodenschnittstelle dieser Klasse definiert neben oberflächenbezogenen Methoden auch das Grundgerüst zur Erkennung und Überprüfung, bzw. zum Sperren oder Freigeben von Benutzereingaben.

Als Trägermedium der Oberflächenbausteine fungieren sogenannte Bildschirmformulare. Deren Aufgabe ist neben der Verwaltung von Oberflächenbausteinen die Bereitstellung des zu ihrer Anzeige benötigten Bildschirmfensters.Dieses ist in folgende Bereiche unterteilt:

- Der Kopf des Fensters besteht aus der Titelleiste, in der die Überschrift des Formulars ausgegeben wird.
- Den Rumpf des Fensters bildet ein für die Ein- bzw. Ausgabefelder des Formulars reservierter Bereich.
- Die Fußleiste enthält zusätzliche, das gesamte Formular betreffende Bedienelemente. Ein mit der Aufschrift „verlassen" versehener Knopf dient zum Schließen des Formulars und ist sowohl im Anzeige- als auch im Eingabemodus des Formulars aktiv. Die Knöpfe „sichern" zum Übernehmen der Benutzer-

eingaben und „zurücksetzen" zur Reinitialisierung des Formulars sind dagegen nur im Editiermodus verfügbar. Um Fehleingaben zu vermeiden oder auf ungesicherte Änderungen des Formularinhaltes hinzuweisen, wurden die Knöpfe „sichern" und „verlassen" mit entsprechenden Sicherheitsmechanismen gekoppelt, die in derartigen Fällen mit dem Benutzer in Dialog treten.

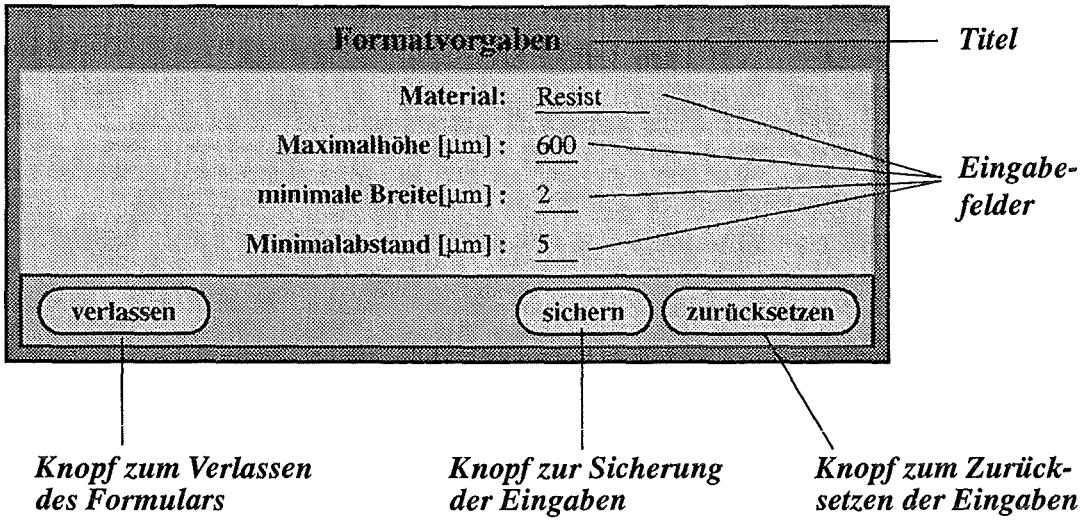


Abbildung 61: Beispiel für ein Bildschirmformular

D.2 Visualisierungsklassen

Die bei MIDAS zur Visualisierung verwendeten Klassen sind in Abb.62 in der in [CY91] verwendeten Notation dargestellt. Aus Platzgründen wurden hierbei nur die wichtigsten Methoden und Attribute der Klassen berücksichtigt.

D.2.1 Die Klasse DisplayForm

Die Klasse DisplayForm stellt dem Anwender den Rahmen eines Eingabeformulars zur Verfügung. hierbei handelt es sich um ein Bildschirmfenster, das als Träger verschiedener Oberflächenbausteine zur Anzeige oder Bearbeitung von Informationen unterschiedlichen Datentyps fungiert. Die Verwaltung der Oberflächenbausteine wird über eine einfach verkettete Liste von Instanzen der Klasse DisplayItem abgewickelt.

D.2.2 Die Klasse DisplayItem

Bei der Klasse DisplayItem handelt es sich um die abstrakte Basisklasse einer auf der XView-Oberfläche basierenden Ein-/Ausgabenklassenhierarchie. Die Unterklassen von DisplayItem stellen Oberflächenbausteine zur Verfügung, mit denen jeweils Informationen eines bestimmten Datentyps je nach Arbeitsmodus der Oberfläche entweder angezeigt oder bearbeitet werden können.

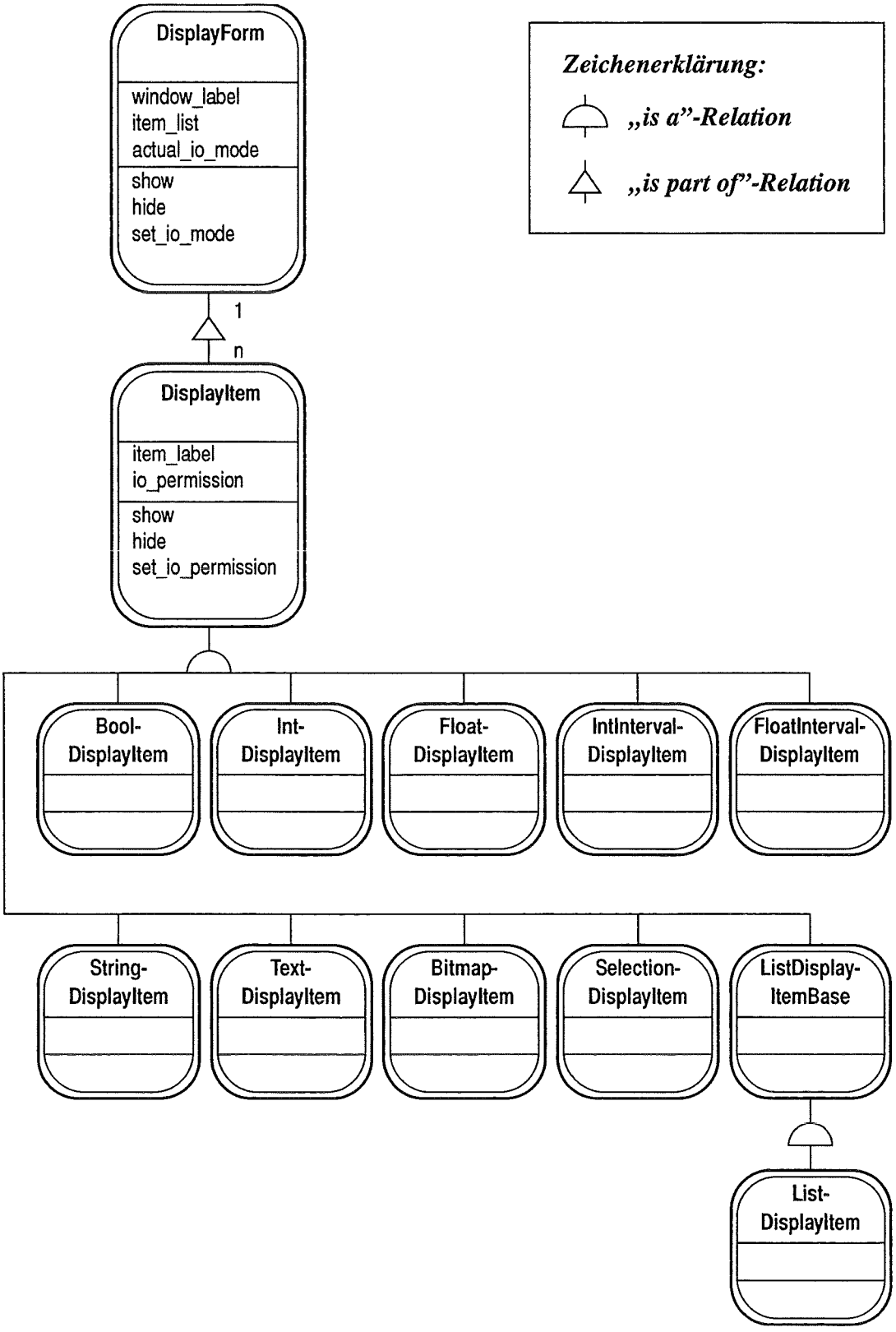


Abbildung 62: Visualisierungsklassen

Um eine Ankopplung eines Oberflächenelementes an beliebige Objekte anderer Klassen zu ermöglichen, kann diesem außerdem eine benutzerdefinierte Aktualisierungsfunktion zugewiesen werden, die bei jeder Sicherung der Benutzereingaben aufgerufen wird.

D.2.3 Die Klasse BoolDisplayItem

Die Klasse BoolDisplayItem stellt einen Oberflächenbaustein zur Verfügung, mit dem der Wert einer Boole'schen Variable abhängig vom gewählten Arbeitsmodus angezeigt und bearbeitet werden kann. Zur Darstellung und Auswahl des Boole'schen Wertes wird ein Schalterpaar verwendet, das die Zustände TRUE und FALSE repräsentiert. Die Beschriftung der Schalter kann vom Benutzer vorgegeben werden.

D.2.4 Die Klasse IntDisplayItem

IntDisplayItem stellt einen Oberflächenbaustein zur Verfügung, mit dem der Wert einer ganzzahligen Varibable abhängig vom gewählten Arbeitsmodus angezeigt und bearbeitet werden kann. Der zulässige Wertebereich der Variablen kann vom Anwender vorgegeben werden.

Das zur Auswahl des Zahlenwertes verwendete Oberflächenelement ermöglicht dem Benutzer, diesen Wertebereich über Maussteuerung zu durchlaufen oder alternativ einen Wert über Tastatur einzugeben.

D.2.5 Die Klasse FloatDisplayItem

Die Klasse FloatDisplayItem stellt einen Oberflächenbaustein zur Verfügung, mit dem der Wert einer Fließkommazahlvaribablen abhängig vom gewählten Arbeitsmodus angezeigt und bearbeitet werden kann. Der zulässige Wertebereich der Variablen und das zu verwendende Zahlenformat können vom Anwender der Klasse vorgegeben werden.

Das Einlesen eines Zahlenwertes erfolgt über die Tastatur.

D.2.6 Die Klasse IntIntervalDisplayItem

Die Klasse IntIntervalDisplayItem stellt einen Oberflächenbaustein zur Verfügung, mit dem ein Ganzzahlintervall abhängig vom gewählten Arbeitsmodus angezeigt und bearbeitet werden kann.

Das hierbei verwendete ermöglicht es dem Benutzer, ein Intervall gemäß der in der Mathematik üblichen Klammernotation einzugeben.

Syntax:

Intervall:
linke Klammer Untergrenze , Obergrenze rechte Klammer

linke Klammer:
] Untergrenze gehört nicht mehr zum Intervall
[Untergrenze gehört zum Intervall
rechte Klammer:
[Obergrenze gehört nicht mehr zum Intervall
] Obergrenze gehört zum Intervall
Untergrenze:
ganzzahliger Wert

Obergrenze:
ganzzahliger Wert

D.2.7 Die Klasse FloatIntervalDisplayItem

Die Klasse FloatIntervalDisplayItem stellt einen Oberflächenbaustein zur Verfügung, mit dem ein reelllzahliges Intervall abhängig vom gewählten Arbeitsmodus angezeigt und bearbeitet werden kann.

Das hierbei verwendete Oberflächenelement ermöglicht es dem Benutzer, ein Intervall gemäß der in der Mathematik üblichen Klammernotation einzugeben.

Syntax:

Intervall:
linke Klammer Untergrenze , Obergrenze rechte Klammer

linke Klammer:
] Untergrenze gehört nicht mehr zum Intervall
[Untergrenze gehört zum Intervall
rechte Klammer:
[Obergrenze gehört nicht mehr zum Intervall
] Obergrenze gehört zum Intervall
Untergrenze:
Fließkommazahl

Obergrenze:
Fließkommazahl

D.2.8 Die Klasse StringDisplayItem

Die Klasse StringDisplayItem stellt einen Oberflächenbaustein zur Verfügung, mit dem eine Zeichenkette abhängig vom gewählten Arbeitsmodus entweder nur angezeigt oder über Tastatur bearbeitet werden kann. Die maximale Länge der Zeichenkette und die bei der Eingabe zugelassenen Zeichen können vom Anwender vorgegeben werden. Unterliegt die einzugebende Zeichenkette komplexeren syntaktischen Randbedingungen, besteht die Möglichkeit, deren Einhaltung über eine vom Anwender zu definierende Funktion verfizieren zu lassen.

D.2.9 Die Klasse TextDisplayItem

Die Klasse TextDisplayItem stellt einen Oberflächenbaustein zur Verfügung, mit dem mehrzeilige Texte abhängig vom gewählten Arbeitsmodus entweder nur angezeigt oder über Tastatur bearbeitet werden können.

Der Anwender hat außerdem die Möglichkeit die Anzahl der darzustellenden Textzeilen und -spalten, sowie die Länge des Texteingabepuffers vorzugeben.

D.2.10 Die Klasse BitmapDisplayItem

Die Klasse BitmapDisplayItem stellt einen Oberflächenbaustein zur Verfügung, mit dem Bitmapdateien im PBM-Format geladen und angezeigt werden können. Der Ladevorgang wird durch Anklicken der Bildanzeigefläche in Gang gesetzt, was ein Fenster zur Dateiauswahl aktiviert. Bei Ladeproblemen wird eine entsprechende Fehlermeldung ausgegeben.

D.2.11 Die Klasse SelectionDisplayItem

Die Klasse SelectionDisplayItem stellt einen Oberflächenbaustein zur Verfügung, mit dem der Benutzer eine Auswahl aus mehreren, über eine Liste von Zeichenketten gegebenen Alternativen treffen kann. Das Resultat der Auswahl wird dem Klassenexternen Bezugsrahmen mit Hilfe einer Integer-Variablen, die den Index des ausgewählten Eintrags enthält, übermittelt.

D.2.12 Die Visualisierungsklassen zur Darstellung von Listen

Die Visualisierungsklassen zur Listenbearbeitung geben dem Benutzer die Möglichkeit, interaktiv über die Benutzeroberfläche auf den Inhalt einer einfach verketteten Liste zuzugreifen. Hierfür wird die Liste in Form einer sogenannten Scrolling List ausgegeben. Der Zugriff auf die einzelnen Listenelemente erfolgt mit der Maus. Die zur Bearbeitung der Liste benötigten Operationen wie Generieren, Kopieren, Löschen, Entfernen oder Wiedereinfügen eines Listeneintrags können vom Benutzer über eine Popup-Menü aufgerufen werden.

Um die Aufgabenstellung der Listenverwaltung unabhängig vom Datentyp der zu verwaltenden Listenklasse zu lösen, basiert die Visualisierungsklasse auf einer generischen Listenklasse der Tools.h++ Bibliothek. Um die Anforderungen an die Methodenschnittstelle des in der Liste zu verwaltenden Datentyps möglichst gering zu halten, müssen die zur Bearbeitung der Listeneinträge benötigten Operationen vom Anwender in Form geeignet definierter Funktionen bereitgestellt werden.

Da die Bearbeitung aller Benutzereingaben bei der XView-Oberfläche über sogenannte Callback-Funktionen mit einer vordefinierten Argumentliste abgewickelt wird, ist es nicht möglich, diese als Template-Funktionen zu definieren, d.h. man keine Möglichkeit, Template-abhängigen Programmcode innerhalb dieser Funktionen zu verwenden. Aus diesem Grund wurde die Aufgabe der Listenvisualisierung auf zwei

Klassen verteilt. Die Template-unabhängige Basisklasse ListDisplayItemBase definiert alle zur Kommunikation mit der XView-Oberfläche benötigten Methoden. In der Unterklasse ListDisplayItem<T> werden diese Methoden zwar Template-abhängig redefiniert, können aber unter Ausnutzung von Polymorphismus über die nicht-generische Basisklasse aufgerufen werden. Hierdurch sind sie auch innerhalb der Template-unabhängigen XView-Callback-Funktionen aufrufbar.

E Die Verwaltung persistenter Daten

E.1 Das Persistenzkonzept von MIDAS

Für eine dauerhafte Sicherung des akquirierten Fertigungswissen benötigt man Methoden, um die im Speicher des Rechners erzeugten, transienten Objekte auf ein externes Speichermedium abzubilden, d.h. sie in persistente Objekte zu überführen.

Während bei transienten Objekten die Vergabe von eindeutigen Objektbezeichnern und die Verwaltung von Speicherressourcen im Regelfall weitgehend über die der Implementierungssprache eigenen Verwaltungsmechanismen abgewickelt werden, benötigt man zur Verwaltung persistenter Objekte zusätzliche Werkzeuge wie z.B. objektorientierte Datenbanken, um eine konsistente Datenhaltung zu gewährleisten. Bei der ersten prototypischen Implementierung verwendet MIDAS hierfür eigene auf Funktionen des UNIX-Dateisystems basierende Verwaltungsmechanismen.

Der Grundgedanke des diesen zugrundeliegenden Persistenzkonzeptes ist, die transiente Datenstruktur auf eine entsprechend modellierte Dateistruktur abzubilden. Im Falle des für MIDAS entwickelten Datenmodells hat man es verschiedenen Klassen von Fertigungswissen und deren Instanzen zu tun, die sich gemäß einem von der Wissensklasse vorgegebenem Schema aus einer Menge verschiedener Attribute zusammensetzen.

Das Persistenzkonzept von MIDAS beruht auf der Idee, daß jedes Objekt der Wissensbasis über entsprechende Methoden Lese-, bzw. Schreibzugriff auf die I/O-Streams hat. In Kombination mit den Visualisierungsmechanismen verfügt man damit bereits über die grundlegende Funktionalität, die zum Editieren einer Wissensbasis auf Schema- bzw. Objektebene benötigt wird.

Um die als Informationsträger konzipierten Objekte von Verwaltungsaufgaben und unter dem Gesichtspunkt der Portabilität vor allem auch von plattformabhängiger Funktionalität des Systems zu entkoppeln, wurden die zur persistenten Speicherung des Fertigungswissens benötigten Datenstrukturen und Methoden in eigenen Klassen gekapselt. Für die Verwaltung der zu einer Wissensklasse gehörenden Informationen wurde eine Klasse SchemeEntry entwickelt, deren Instanzen im weiteren als Schemaeintrag bezeichnet werden sollen. Ihre Aufgabe besteht in der persistenten Speicherung des Attributschemas einer Wissensklasse. Für die Persistenz von Objekten sind sogenannte Objekteinträge verantwortlich, bei denen es sich um Instanzen der Klasse ObjectEntry handelt.

Die wesentliche Aufgaben dieser zusätzlichen Verwaltungsklassen besteht darin, eine konsistente Haltung der persistenten Daten zu gewährleisten. Eine der wichtigsten Voraussetztungen hierfür ist die Verwendung eindeutiger Bezeichner für die zu verwaltenden Objekte. Unter dem Blickwinkel des semantischen Bezuges eines Objektes, bietet sich hierfür zunächtst ein entsprechend vom Benutzer definierte Objekt-, bzw. Klassenname an. Hierbei tritt allerdings das Problem auf, daß diese

benutzerdefinierten Namen nicht notwendigerweise das Kriterium der Eindeutigkeit erfüllen, sofern bei der Namensdefinition bzw. bei Eingaboperationen zur Änderung eines Namens nicht zusätzliche Kontrollmechanismen eingeschaltet werden.

Aus diesem Grund basiert bei MIDAS die Identifikation von Objekt- und Schemaeinträgen auf der Vergabe von Kennziffern, die während deren gesamter Lebenszeit unveränderlich bleiben. Die Namen dienen dem Benutzer jedoch weiterhin als Schlüssel für den Zugriff auf ein Attributschema oder Objekt und werden bei Eingabeoperationen auf ihre Eindeutigkeit überprüft.

Die Schnittstelle zwischen Kennziffern, den dazugehörigen Sicherungsdateien und benutzerdefinierten Namen bilden sogenannte Indextabellen. Jedes Attributschema und Objekt ist in einem entsprechenden Index erfaßt. Alle Anfragen nach bestimmten Einträgen und gegebenfalls notwendige Zugriffe auf die Datenbasis werden über diese Indextabellen abgewickelt.

Schema- und Objektindex sind selbst Bestandteil eines übergeordneten Verwaltungsobjektes. Bei diesem handelt es sich um den sogenannten Wissensmanager, dessen Aufgabe in der Integration der vorgestellten Verwaltungsklassen in die Oberfläche des Informationssystems besteht.

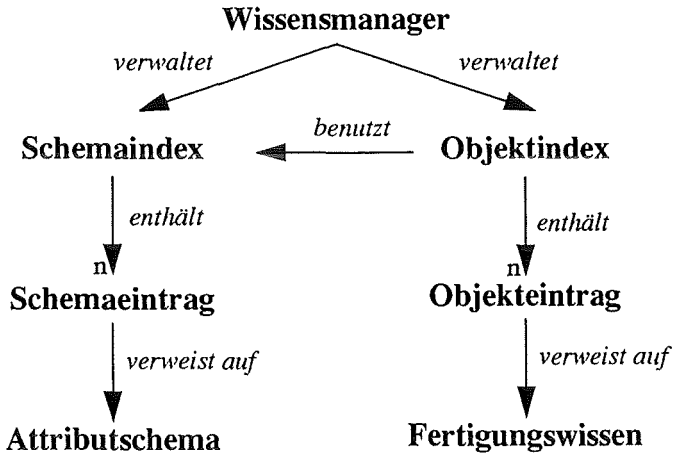


Abbildung 63: Persistenzkonzept von MIDAS

E.2 Zur Verwaltung persistenter Daten entwickelte Klassen

Die zur Realisierung des vorgestellten Persistenzkonzeptes entwickelten Klassen zeigt Abb.64 in der in [CY91] üblichen Notation. Aus Platzgründen wurden hierbei jeweils nur die wichtigsten Attribute und Methoden berücksichtigt.

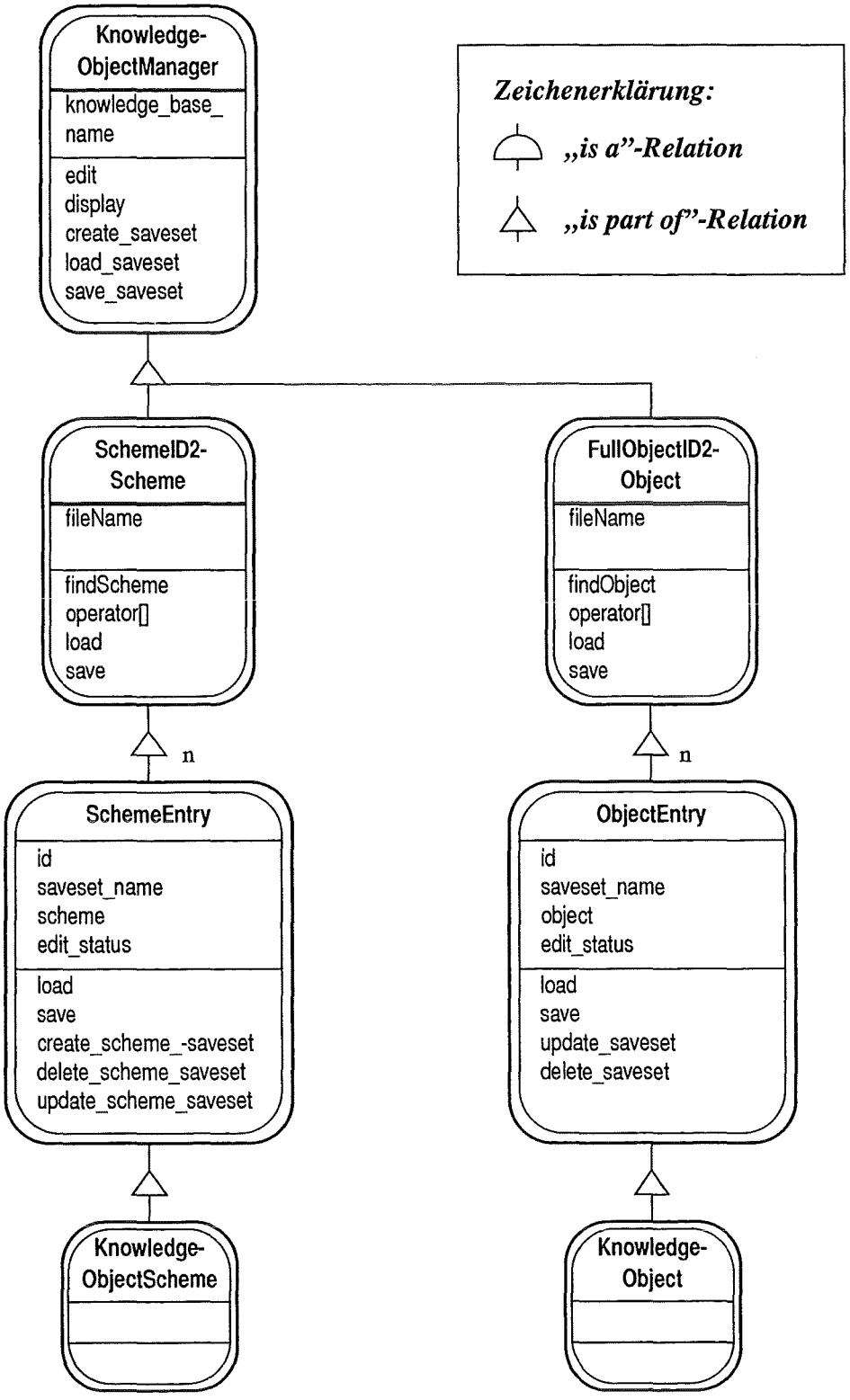


Abbildung 64: *Klassen zur Verwaltung persistenter Daten*

E.2.1 Die Klasse KnowledgeObjectManager

Aufgabe der Klasse KnowledgeObjectManager ist die Verwaltung persistenter Attributschemata und Objekte in Verbindung mit einer Benutzeroberfläche. Die Speicherung der Daten wird zum gegenwärtigen Zeitpunkt mit Hilfe der Klassen SchemeID2Scheme und FullObjectID2Object über das UNIX-Dateisystem abgewikkelt.

Die Oberfläche des Persistenzmanagers besteht im wesentlichen aus zwei maussensitiven Listen. In der ersten Liste werden die vorhandenen Attributschemata angezeigt. Wird ein Schema selektiert, erscheinen alle zu diesem Schema gehörigen Objekte in der zweiten Liste.

Beide Listen können - sofern die Eingabe freigegeben ist - vom Benutzer bearbeitet werden.

E.2.2 Die Klasse SchemeEntry

Die Klasse SchemeEntry hat die Aufgabe, die als Träger von Schemainformation konzipierten Instanzen der Klasse KnowledgeObjectScheme von Verwaltungsfunktionen, insbesondere der Problematik der Persistenz zu enkoppeln. Hierzu wird eine Instanz der Klasse KnowledgeObjectScheme in eine Instanz der Klasse SchemeEntry eingebettet, die eine Schnittstelle zu den Verwaltungsmechanismen der Wissensbasis bereitstellt.

Die Speicherung einer Instanz der Klasse SchemeEntry und aller von ihr abhängigen Objekte wird über das UNIX-Dateisystem abgewickelt. Hierbei legt die Instanz ein eigenes Verzeichnis an, in welchem Schemainformation, eine Tabelle mit allen dazugehörigen Wissensbereichen und deren jeweilige Sicherungsdateien abgelegt werden. Als Verzeichnisnamen wird eine Zeichenkette verwendet, die sich aus dem Buchstaben „S“ (=scheme) und einer eindeutigen Identifikationsnummer in hexadezimaler Notation zusammensetzt.

E.2.3 Die Klasse SchemeID2Scheme

Die Klasse SchemeID2Scheme dient der Verwaltung der in einer Wissensbasis erfaßten Schemaeinträge in Form einer Indextabelle. Die Identifikation der Schemaeinträge erfolgt mit Hilfe einer eindeutigen Schemakennzahl.

E.2.4 Die Klasse ObjectEntry

Die Klasse ObjectEntry hat die Aufgabe, die als Wissensträger konzipierten Instanzen der Klasse KnowledgeObject von Verwaltungsfunktionen, insbesondere der Problematik der Persistenz zu entkoppeln. Hierzu wird eine Instanz der Klasse KnowledgeObject in eine Instanz der Klasse ObjectEntry eingebettet, die eine Schnittstelle zu den Verwaltungsmechanismen der Wissensbasis bereitstellt. Die Identifizierung der Objekte wird über die Vergabe eindeutiger IDs abgewickelt.

Zur persistenten Speicherung eines Objektes wird eine Sicherungsdatei angelegt. Deren Namen setzt sich aus dem Buchstaben „K“ und der Objekt-ID in hexadezimaler Notation zusammen.

E.2.5 Die Klasse FullObjectID2Object

Die Klasse FullObjectID2Object dient der Verwaltung der in einer Wissensbasis erfaßten Objecteinträge in Form einer Indextabelle. Die Identifikation der Objekteinträge erfolgt mit Hilfe einer eindeutigen Objektkennzahl, in der auch die Kennzahl des dazugehörigen Schemaeintrages enthalten ist.

F Bei Midas verwendete Dateiformate

F.1 Beim Informationssystem verwendete Datenformate

F.1.1 Format von Namen

Klassen-, Objekt- und Attributnamen werden bei MIDAS als in Anführungs- und Schlußzeichen eingeschlossener Text abgespeichert. Der Text darf aus Groß- und Kleinbuchstaben, Ziffern und Leerzeichen bestehen.

Syntax:

Name:
" *Text* "

Text:
a..z, A..Z, 0..9, <SPC>

F.1.2 Format der Klassenindexdatei

Eine Indexdatei, in der *n* Wissensklassen erfaßt sind, besitzt folgenden Aufbau:

Syntax der Datei	Erklärung
n	Anzahl der Wissensklassen
Klassenname #1 ID#1<EOL>	Name und Kennziffer von Klasse #1
Klassenname #2 ID#2<EOL>	Name und Kennziffer von Klasse #2
:	:
Klassenname #n ID#n<EOL>	Name und Kennziffer von Klasse #n
<EOF>	Dateiende

ID:
Integer-Zahl

F.1.3 Format einer Klassenschemadatei

Ein Klassenschema, das n Attributkonfigurationen enthält, wird in folgendem Format gesichert:

Syntax der Schemadatei	Erklärung
Klassenname n	Klassenname, Anzahl der Attributkonfigurationen
Attributtyp#1 Attributkonfiguration#1<EOL>	Attributtyp#1, typspezifische Konfigurationsdaten

Attributtyp#2 Attributkonfiguration#2<EOL>	Attributtyp#2, typspezifische Konfigurationsdaten
Attributtyp#n Attributkonfiguration#1<EOL>	Attributtyp#2, typspezifische Konfigurationsdaten

<EOF>

Hierbei sind folgende Attributtypen vorgesehen:

Syntax:	**Erklärung**
Attributtyp:	
0	Bool-Attribut*)
1	Integer-Attribut
2	Float-Attribut
3	String-Attribut
4	Text-Attribut
5	Link-Attribut, Verweis auf ein Objekt*
6	Dokument-Attribut*)
7	Bitmap-Attribut

Die mit *) gekennzeichneten Attributtypen wurden bislang nicht implementiert.

F.1.4 Format von Attributkonfigurationen

Derzeit können Attributkonfigurationen für ganze und reelle Zahlen, Zeichenketten, Texte und Bilddateien im PBM-Format vorgegeben werden.

Attributkonfiguration:
Integer-Konfigurationsdaten
Float-Konfigurationsdaten
String-Konfigurationsdaten
Text-Konfigurationsdaten
Bitmapdatei-Konfigurationsdaten

F.1.5 Format einer Integer-Attributkonfiguration

Die Konfiguration eines Integer-Attributes beinhaltet den Attributnamen, den maximalen bzw. minimalen Attributwert, sowie einen Defaultwert.

Integer-Konfigurationsdaten:
Attributname Integer-Minimalwert Integer-Maximalwert Integer-Defaultwert

Integer-Minimalwert:
Integer-Wert

Integer-Maximalwert:
Integer-Wert

Integer-Defaultwert:
Integer-Wert

F.1.5.1 Format einer Float-Attributkonfiguration

Über die Konfiguration eines Float-Attributes werden Attributname, maximaler und minimaler Wert, sowie ein Defaultwert vorgegeben.

Float-Konfigurationsdaten:
Attributname Float-Minimalwert Float-Maximalwert Float-Defaultwert

Float-Minimalwert:
Float-Wert

Float-Maximalwert:
Float-Wert

Float-Defaultwert:
Float-Wert

F.1.5.2 Format einer String-Attributkonfiguration

Mithilfe einer String-Attributkonfiguration werden der Attributname, ein Eingabefilter, die maximale Länge der Zeichenkette und der Defaultwert festgelegt. Als Eingabefilter dient eine sogenannte „single character expression" im UNIX-üblichen Format [Sun93].

String-Konfigurationsdaten
Attributname String-Eingabefilter String-Länge String-Defaultwert

String-Eingabefilter:
" *single-character-expression* "

String-Länge:
Integer-Wert

String-Defaultwert:
" *Textzeichen* "

F.1.5.3 Format einer Text-Attributkonfiguration

Die Konfiguration eines Textattributes beinhaltet den Attributnamen, die Anzahl der darzustellenden Textzeilen und -spalten, sowie die Länge des bereitzustellenden

Textspeichers.

Syntax:
Text-Konfigurationsdaten
Attributname Textzeilen Textspalten Textlänge

Textzeilen:
Integer-Wert

Textspalten:
Integer-Wert

Textlänge:
Integer-Wert

F.1.5.4 Format einer Bimap-Attributkonfiguration

Die Konfiguration eines Bitmapattributes enthält den Attributnamen, sowie die Anzahl der darzustellenden Beildzeilen und -spalten.

Syntax:
Bitmap-Konfigurationsdaten
Attributname Bildzeilen Bildspalten

Bildzeilen:
Integer-Wert

Bildspalten:
Integer-Wert

F.1.6 Format einer Objektindexdatei

Eine Indexdatei, in der *n* Objekte erfaßt sind, besitzt folgenden Aufbau:

Syntax der Indexdatei	**Erklärung**
n	Anzahl der erfaßten Objekte
Objektname #1 ID#1<EOL>	Name und Kennzahl von Objekt#1
Objektname #2 ID#2<EOL>	Name und Kennzahl von Objekt#2
:	:
Objektname #n ID#n<EOL>	Name und Kennzahl von Objekt#n
<EOF>	Dateiende

F.1.7 Format der Sicherungsdatei eines Objektes

Ein aus n Attributen bestehendes Objekt wird in folgendem Format gesichert:

Syntax der Objektdatei	Erklärung
Objektname n<EOL>	Name des Objektes und Anzahl der Attribute
Attributwert#1<EOL>	Attributwert#1, typabhängig
Attributwert#2<EOL>	Attributwert#2, typabhängig
:	:
Attrubutwert#n<EOL>	Attributwert#n, typabhängig
<EOF>	

Attributwert:
- *Bool-Wert*
- *Integer-Zahl*
- *Float-Attribut*
- *String*
- *Text*
- *Dateinamen*

F.2 Im Rahmen des Diagnosesystems verwendete Datenformate

F.2.1 Dateischnittstelle des Moduls zur Regelselektion

Das Modul zur Regelselektion verwaltet Sätze von Entwurfsregeln, die in Form einer Sequenz von einzelnen Regeln strukturiert sind. Die Datei eines aus n Regeln bestehnden Regelsatzes ist dieser sequentiellen Struktur entsprechend folgendermaßen strukturiert:

Syntax	Bedeutung
Regelsatzname n<EOL>	Name des Regelssatzes, Anzahl der darin enthaltenen Regeln
ObjektID#1<EOL>	Objektkennzahl von Regel#1
ObjektID#2<EOL>	Objektkennzahl von Regel#2
: :	:
ObjektID#n<EOL>	Objektkennzahl von Regel#n
<EOF>	Dateiende

F.2.2 Dateiformat eines Merkmalssatzes

Bei einem Merkmalssatz hat man es dementsprechend mit einer Liste von Merkmalen unterschiedlichen Datentyps zu tun. Zu deren Speicherung wird bei MIDAS folgendes Dateiformat verwendet.

n<EOL>

Parametertyp-ID#1 Merkmalsname#1 Lokation#1 Datentyp#1 Parameterwert#1<EOL>
Parametertyp-ID#2 Merkmalsname#2 Lokation#2 Datentyp#2 Parameterwert#2<EOL>
:
Parametertyp-ID#n Merkmalsname#n Lokation#n Datentyp#n Parameterwert#n<EOL>
<EOF>

Parametertyp-ID:
0 (geometriebezogener Parameter)
1 (prozeßbezogener Parameter)
2 (materialbezogener Parameter)

Merkmalsname:
"Textzeichen"

Lokation:
Integer-Wert

Datentyp:
0 (Bool)
1 (Integer)
2 (Float)

Parameterwert:
Bool-Wert
Integer-Wert
Float-Wert

F.2.3 Dateiformat eines Fehlerprotokolls

Die Sicherung eines Fehlerprotokolls, in dem n fehlerhafte Geometriemerkmale erfaßt sind, erfolgt in folgendem Format:

n<EOL>
Parametertyp-ID#1 Merkmalsname#1 Lokation#1 Datentyp#1 Parameterwert#1<EOL>
n_1<*EOL*>
ObjektID#1<EOL>
: :
ObjektID#n_1<EOL>
Parametertyp-ID#2 Merkmalsname#2 Lokation#2 Datentyp#2 Parameterwert#2<EOL>
n_2<*EOL*>
ObjektID#1<EOL>
: :
ObjektID#n_2<EOL>
*Parametertyp-ID#*n *Merkmalsname#n Lokation#n Datentyp#n Parameterwert#n*<EOL>
n_n<*EOL*>

ObjektID#1<EOL>
: :
ObjektID#n_n<EOL>
<EOF>

Literaturverzeichnis

Beck82 E.W. BECKER, H. BETZ, W. EHRFELD, W. GLASHAUSER, A. HEUBERGER, H.J. Michel, D. Münchmeyer, S. Pongratz, R. v. Siemens, *Naturwissenschaften 69,* 1982, S. 520

BGH91 B. BÜRG, H. GUTH, A. HELLMANN: *Parametric Optical Measurements of Micromechanical Structures with Arbitrary Plane Surface Geometries: The COSMOS-2D System*", R. Krahn (Hrsg.), Micro System Technologies 91: 2. Internat Fachkongress, VDE Verlag Berlin, Offenbach, 1991, S.280-287.

BKS95 R. BRÜCK, K. HAHN, J. STIENECKER: *Technology description methods for LIGA processes*, J. Micromech. Microeng. 5 (1995), S.196-198

Bode88 K.H. BODE: *Konstruktionsatlas*, Hoppenstett Technik Tab. Verlag, Darmstadt 1988

Bra94 I. H. BRAUCH: *Wissensbasierte Modellierung des LIGA-Fertigungsprozeßes*, Dissertation, Fakultät für Maschinenbau, Universität Karlsruhe,1994

Bro82 I.N. BRONSTEIN, K.A. SEMENDJAJEW: *Taschenbuch der Mathematik*, 22. Auflage, Verlag Harri Deutsch, Thun / Frankfurt a.M.,1982, S.325

Brü93 R. BRÜCK: *Entwurfswerkzeuge für VLSI-Layout*, Carl Hanser Verlag, München/ Wien 1993

BuSh84 B.G. BUCHANAN, E.H. SHORTLIFFE: *The MYCIN Experiments of the Stanford Heuristic Programming Project,* Addison-Wesley Publishing Company, Reading/Massachusetts, 1984

Cal84 CALMA COMPANY: *Stream Format GDS II, Release 5.1*, 1984

CY91 P.COAD, E. YOURDON: *Object-oriented Analysis*, Prentice Hall, 1991

EB90 W. EHRFELD, E.W. BECKER: *Das LIGA-Verfahren zur Herstellung von Mikrostrukturkörpern mit großem Aspektverhältnis und großer Strukturhöhe,* KfK-Nachrichten 4/1987, S. 167-179

Egg95 H. EGGERT, C. DÜPMEIER, H. GUTH, K.P. SCHERER, W. SÜß: *Eine Informationsverarbeitungsumgebung zur Konstruktion und Vermessung von Mikrostrukturen,* Vortragsveranstaltung: *Informationstechnik für Mikrosysteme,* VDE/VDI-Gesellschaft Mikroelektronik (GME), Stuttgart-Büsenau, 17. Januar 1996, Tagungsband S.53-57

ESS94 H. EGGERT, K.P. SCHERER, P. STILLER: *Geometriewissensbasis als Schnittstelle MCAD/ECAD,* W.John, H.Eggert (Hrsg.), 1. Workshop *Methoden- und Werkzeugentwicklung für den Mikrosystementwurf,* Karlsruhe, 15. November 1994, Tagungsband S.94-101

FFL94 B. FORNER, R. FEIERTAG, C. LEßMÖLLMANN: *Wissensabfrage zur Unterstützung der Konstrution von LIGA-Mikrostrukturen,* W.John, H.Eggert (Hrsg.),1. Workshop *Methoden- und Werkzeugentwicklung für den Mikrosystementwurf,* Karlsruhe, 15. November 1994, Tagungsband S.109-116

FFR95 R. FEIERTAG, B. FORNER, A. ROGNER: *Konstruktionsunterstützung für LIGA-*

Mikrostrukturen unter fertigungs- und montagegerechten Aspekten, W. John, H. Eggert K.D. Müller-Glaser (Hrsg.): 2. Workshop *Methoden- und Werkzeugentwicklung für den Mikrosystementwurf*, Karlsruhe, 14. November 1995, Tagungsband S.64-71

FK85 R. FIKES, T. KEHLER: *The Role of Frame-Based Representation in Reasoning*, Communcations of the ACM, Vol. 28, September 1994, S.904-920

GDGHS93 H. GUTH, C. DÜPMEIER, M. GOIK, A. HELLMANN, U. STUCKY: *Automatische Vermessung von 2D und 3D-LIGA-Strukturen zur Qualitätskontrolle*, 1. Statuskolloquium des Projektes Mikrosystemtechnik, Kernforschungszentrum Karlsruhe, KfK 5238, 1993, Tagungsband S.176-182

Gul88 J. GULBINS: *UNIX - Version 7, bis System V.3*, Springer Verlag, 2. Auflage, New York / Heidelberg / Berlin / Tokyo, 1988

Hahn95 K. HAHN: *LIDO - Entwurfsregelprüfung für LIGA-Maskenlayouts*, Forschungsbericht 577, Fachbereich Informatik, Universität Dortmund, 1995

HaLe90 D.HARTMANN, K.LEHNER: *Technische Expertensysteme*, Springer-Verlag, Berlin/Heidelberg/New York, 1990

Hei92 H. HEIN, P. BLEY, J. GÖTTERT, U. KLEIN: *VDI Berichte 960, VDI-Verlag, Düsseldorf, 1992, S. 75*

HK85 P. HARMON, D. KING: *Expert Systems - Artificial Intelligence in Business*, John Wiley & Sons, Inc., New York, 1985

Hel91 D. HELLER: *XView Programming Manual for XView Version 3.0*, O'Reilley & Associates, Inc., 3. Auflage, September 1991

IGES91 IGES/PDES ORGANIZATION: *The Initial Graphics Exchange Specification (IGES), Version 5.1*, 1991

Ill92 J.A. ILLIK: *Programmieren in C unter UNIX*, Sybex Verlag, 3. Auflage, Düsseldorf, 1992

KKP87 D. KARRAS, L. KREDEL, U. PAPE: *Entwicklungsumgebungen für Expertensysteme*, de Gruyter, Berlin, 1987

KR90 B.W. KERNIGHAN, D.M. RITCHIE: *Programmieren in C*, Carl Hanser Verlag, 2. Auflage, München/Wien,1990

Leß92 C. LEßMÖLLMANN: *Fertigungsgerechte Gestaltung von Mikrostrukturen für die LIGA-Technik*, Dissertation, Universität Karlsruhe 1992

MB93 W. MENZ, P. BLEY: *Mikrosystemtechnik für Ingenieure*, VCH Verlagsgesellschaft mbH, Weinheim 1993

Mey88 B. MEYER (Hrsg.): *Objektorientierte Softwareentwicklung*, Carl Hanser Verlag, München/Wien 1988

Mohr88 J. MOHR, W. EHRFELD, D. MÜNCHMEYER: *Analyse der Defektursachen und der Genauigkeit der Strukturübertragung bei der Röntgentiefenlithographie mit Synchrotronstrahlung*, Dissertation J. Mohr, Kernforschungszentrum Karlsruhe, KfK 4414, 1988

PaBe86 G. PAHL, W. BEITZ: *Konstruktionslehre*, Springer Verlag, Heidelberg, 1986

RWS94 ROGUE WAVE SOFTWARE, INC.: *Tools.h++ Introduction and Reference Manual*, Corvalis/Oregon, 1994

SBESS95 K.P. SCHERER, P. BUCHBERGER, H. EGGERT, P. STILLER, U. STUCKY: *Gesamtarchitektur und Schemaentwicklung als Basis zur Konstruktionsunterstützung*, W. John, H. Eggert K.D. Müller-Glaser (Hrsg.), 2. Workshop *Methoden- und Werkzeugentwicklung für den Mikrosystementwurf*, Karlsruhe, 14. November 1995, Tagungsband S.56-63

SEBSW94 K.P. SCHERER, H. EGGERT, P. BUCHBERGER, P. STILLER, P. WIELAND: *Wissensbasierte Entwurfsunterstützung zur Herstellung von LIGA-Mikrostrukturen*, W. John, H. Eggert (Hrsg.), 1. Workshop *Methoden- und Werkzeugentwicklung für den Mikrosystementwurf*, Karlsruhe, 15. November 1994, Tagungsband S.156-163

Str92 B. STROUSTRUP: *Die C++ Programmiersprache*, 2. Auflage, Addison-Wesley Publishing Company (Deutschland), 1992

Sun93 SUNSOFT: *Solaris 2.3 Benutzerhandbuch*, Mountain View/CA, 1993

Sun94a SUNSOFT: *OpenWindows Developer's Guide: User's Guide*, Mountain View/CA, 1994

Sun94b SUNSOFT: *OpenWindows Developer's Guide: XView Code Generator Programmer's Guide*, Mountain View/CA, 1994

ThSc89 N.H.C. THUY, P. SCHNUPP: *Wissensverarbeitung und Expertensysteme*, Oldenbourg Verlag, München /Wien 1989

Tra93 H. TRAUBOTH: *Aufgaben der Informationsverarbeitung in der Mikrosystemtechnik*, 1. Statuskolloquium des Projektes Mikrosystemtechnik, Kernforschungszentrum Karlsruhe, KfK 5238, 1993, S.36-47

VR91 T. VAN RAALTE (Hrsg.): *XView Reference Manual for XView Version 3*, O'Reilley & Associates, Inc., September 1991

Wat86 D.WATERMAN: *A Guide to Expert Systems*, Addison-Wesley Publishing Company, Reading/Massachusetts, 1986

Wie95 P. WIELAND: *Entwurf eines Systems zur Erfassung und Weiterverarbeitung von Produktinformation bei der Herstellung von Mikrostrukturen*, Dissertation, Fakultät für Maschinenbau, Universität Karlsruhe, 1995

Danksagung

An dieser Stelle möchte ich die Gelegenheit nutzen, den vielen Leuten meinen Dank auszusprechen, die auf vielfältige Weise zum Gelingen dieser Arbeit beigetragen haben.

Für die Übernahme des Referats und sein Engagement bei der Durchführung des Promotionsverfahrens möchte ich Herrn Prof. Dr. Laur herzlich danken und ebenso Herrn Prof. Dr. Benecke für die Übernahme des Korreferates.

Die vorliegende Arbeit entstand in den Jahren 1993 bis 1996 am Institut für Angewandte Informatik des Forschungszentrums Karlsruhe in Zusammenarbeit mit dem Institut für Mikrostrukturtechnik und wurde den überwiegenden Teil dieser Zeit von Herrn Prof. Dr. Menz und Herrn Prof. Dr. Trauboth betreut, wofür ich beiden meinen Dank ausspreche.

Besonderer Dank gebührt in diesem Zusammenhang auch meinem Abteilungsleiter Herrn Dr. Horst Eggert, der diese Arbeit erst ermöglicht und während der gesamten Zeit mit großem Engagement und Kreativität unterstützt hat.

Außerdem möchte ich allen Mitarbeitern der Abteilung Mikrosysteminformatik für die angenehme, von großer Hilfsbereitschaft geprägte Arbeitsatmosphäre danken, besonders Frau Christina Grieß und meinem Gruppenleiter Herrn Dr. Klaus Peter Scherer, die beide immer ein offenes Ohr für meine Probleme hatten. Ein nicht minder herzliches Dankeschön geht auch an meinen Kollegen Dr. Klaus Lindemann, der mit großem Einsatz an der Implementierung von MIDAS mitgearbeitet und innerhalb kürzester Zeit die Portierung des Systems auf eine andere Benutzeroberfläche bewältigt hat. Für ihre Hilfe bei den verschiedensten technischen Problemen möchte ich mich bei meinen Kollegen Peter Stiller, Dr. Uwe Stucky, Dr. Wolfgang Süß, Dr. Helmut Guth, Dr. Clemens Düpmeier und Herrn Stefan Maihack bedanken. Frau Sabine Scheer, sowie Frau Peters vom Intstitut für Theoretische Elektrotechnik und Mikroelektronik der Universität Bremen danke ich herzlich für ihre tatkräftige Unterstützung in diversen organisatorischen Angelegenheiten.

Vergessen möchte ich an dieser Stelle auf keinen Fall meine Familie und meine Freunde, die mir während dieser Zeit zur Seite gestanden und so auf ihre Weise nicht unerheblich zum Gelingen dieser Arbeit beigetragen haben.