Informatik Forsch. Entw. (1987) 2: 1–18

Informatik
Forschung und Entwicklung

KUNICAD – ein datenbankgestütztes geometrisches Modellierungssystem für Werkstücke

Th. Härder[1], Ch. Hübel[1], St. Langenfeld[2] und B. Mitschang[1]

[1] Universität Kaiserslautern, Fachbereich Informatik, Erwin-Schrödinger-Straße, Gebäude 48, D-6750 Kaiserslautern
[2] Lufthansa AG, Flughafen Frankfurt/Basis, FRA AS 53, D-6000 Frankfurt/Main

Zusammenfassung. Es wird ein datenbankgestütztes, volumenorientiertes Modellierungssystem für Werkstücke beschrieben, das als Kernalgorithmus für die geometrische Modellierung ein volumenorientiertes Verfahren einsetzt. Zur Unterstützung der graphischen Repräsentation der Werkstücke werden intern zusätzlich Strukturen nach dem Begrenzungsflächenmodell gehalten, die automatisch aus den CSG-Strukturen abgeleitet und nachgeführt werden.

Das KUNICAD-System wird durch seine Gesamtarchitektur, seine Benutzerschnittstelle und sein zugrundeliegendes Anwendermodell skizziert. Die Modellierungskomponente wird durch ihre wesentlichsten Aufgaben – die Darstellung der Objekte und ihre Handhabung – beschrieben. Eine objektunterstützende Datenbankschnittstelle wurde nach dem Zusatzebenen-Architekturkonzept auf der Basis eines CODASYL-Datenbanksystems (UDS) entwikkelt. Das implementierte System gestattet das Studium von praxistauglichen CAD-Schnittstellen und erlaubt eine detaillierte Analyse des durch die Datenbankverwaltung bedingten Mehraufwands bei CAD-Operationen.

Schlüsselwörter: Geometrisches Modellieren, CAD-System, Körperdarstellung, Volumenmodell, Begrenzungsflächenmodell, Datenbankunterstützung, Leistungsbewertung

Abstract. A volume-oriented modeling system for solids using a database system for all data management functions is described in detail. It relies on a volume-oriented algorithm for geometric modeling. To support the graphic representation of solids, their boundary representation, automatically derived from CSG structures and maintained, is provided internally.

The KUNICAD system is sketched by its overall architecture, its user interface and the related application model. The modeling component is described by its essential tasks – the representation of objects and their manipulation. Database management functions are enhanced by the development of an object-supporting interface whereby a CODASYL database system (UDS) is extended by an additional layer. The implemented system permits the study of practical CAD interfaces and allows for a detailed analysis of the overhead produced by the database management for CAD operations.

Key words: Geometric modeling, CAD system, solid representation, constructive solid geometry, boundary representation, database support, performance evaluation

CR Subject Classifications: H.2.4, H.4.2, I.3.2, I.3.5, J.6

1. Einleitung

War bis vor kurzem die Datenverarbeitung geprägt von Aufgaben aus dem administrativ-betriebswirtschaftlichen Bereich – besonders bei datenbankgestützten Anwendungen –, so ändert sich das Bild momentan sehr rasch durch „nicht-kommerzielle" Rechneranwendungen, insbesondere aus dem Bereich der Ingenieuraufgaben, der ein breites Spektrum von äußerst heterogenen und vielfältigen Anwendung aufweist. Von besonderer Wichtigkeit ist hierbei die Rechnerunterstützung beim Entwickeln, Entwerfen, Konstruieren und Zeichnen, die unter dem Kürzel CAD (computer aided design) bereits Eingang in unseren Sprachgebrauch gefunden hat. Zu ihren primären Anwendungsgebieten zählen Disziplinen wie Maschinenbau, Elektrotechnik, Bauingenieurwesen und Architektur, in denen immer

noch zu einem großen Teil Produktunterlagen mit Hilfsmitteln wie Bleistift und Zeichenbrett unter hohem Zeitaufwand hergestellt werden [9]. Aus Gründen der Kostenkontrolle, der Konstruktionsökonomie, der einheitlichen Erfassung und Darstellung der Entwurfsdaten sowie ihrer Austauschbarkeit wird in diesen Gebieten ein starker Zwang zur Einführung von CAD-Systemen ausgeübt.

Allein für die Aufgaben des Entwurfs gibt es bis heute schon über 300 CAD-Pakete, die teils die Bearbeitung von zweidimensionalen, teils die von dreidimensionalen Objekten erlauben. Dazu ist eine Abbildung der geometrischen Gebilde auf Datenstrukturen erforderlich, die mangels einer verbindlichen Norm und eines allseitig einsetzbaren Modells wegen der Komplexität der Abbildung sehr unterschiedlich ausfallen kann. Deshalb wurde in den CAD-Systemen die Datenverwaltung fast ausschließlich individuell und auf einer ad-hoc-Basis mit Hilfe von separaten Dateien gelöst. Gravierende Nachteile mußten dabei in Kauf genommen werden [16, 21]. Folge davon sind dringliche Normierungsbemühungen wie IGES [18], um wenigstens durch Schnittstellendefinition und Konversion einen Teil der Produktdaten austauschen zu können.

DB-gestützte CAD-Systeme sind bisher kaum verfügbar. Dabei würden sie viele der Nachteile von dateiorientierten CAD-Systemen wie übermäßige Redundanz der Daten, fehlende Integritätskontrolle, mangelnde Datenstrukturierung usw. vermeiden sowie Standardisierung, Austausch und Übertragung von komplexen Datenstrukturen verbessern. Außerdem würden durch ein Datenbanksystem (DBS) zusätzliche Aufgaben wie Kontrolle des Mehrbenutzerbetriebs, Sicherung der Daten, Strukturierung der Verarbeitung durch ein Transaktionskonzept usw. übernommen werden. Wesentlicher Hemmschuh einer solchen Entwicklung sind die auf dem Markt verfügbaren kommerziellen DBS, die für die erforderlichen Aufgaben denkbar schlecht gerüstet sind. Das liegt vor allem an ihrem mangelnden Anwendungsbezug bei der Datenmodellierung und an ihren für die vorgegebenen Konstruktionsaufgaben überhaupt nicht geeigneten DB-Operationen.

Die Programmierung von CAD-Algorithmen auf DB-Programmierschnittstellen heutiger DBS (relational, netzwerkorientiert) ist äußerst mühsam und dem Anwender nicht zuzumuten [9, 24]. Besser geeignete DBS sollten deshalb eine abstraktere, stärker objektorientierte Schnittstelle aufweisen, deren Datenstrukturen und Operationen sich durch einen starken Anwendungsbezug auszeichnen. Erst unter dieser Voraussetzung lassen sich CAD-Systeme entwerfen, die ergonomische Benutzerschnittstellen bereitstellen, die sich dem Arbeitsablauf und den Denkgewohnheiten anpassen lassen. Solche Schnittstellen mit interaktiver Graphikunterstützung sind eine wesentliche Voraussetzung für die Akzeptanz bei der Anwendung für kreative Entwurfsarbeiten wie beispielsweise der Konstruktion „körperhafter" Bauteile.

Im Moment verbietet sich eine vollständige Neuentwicklung eines datenbankgestützten CAD-Systems schon allein aus Mangel an DB-Erfahrung im CAD-Bereich und in unserem Fall auch aus Aufwandsgründen. Um aber gerade diese Erfahrung erwerben zu können, eignet sich vor allem eine Prototyp-Entwicklung, die nicht Leistungs- sondern Funktionsaspekte betont. Ausgangspunkt war ein CODASYL-DBS (UDS), das im Sinne der Zusatzebenen-Architektur [16] durch Hinzufügen abstrakterer Schnittstellen erweitert wurde. Ziel war die Entwicklung eines DB-gestützten, volumenorientierten Modellierungssystems für Werkstücke, das ein ausführliches Studium von praxistauglichen CAD-System-Schnittstellen erlaubt und durch eine realitätsnahe Konstruktionsumgebung detaillierte und genaue Aufwandsuntersuchungen für CAD-Operationen unterstützt. Insbesondere ergab sich dafür folgende detailliertere Zielsetzung:

- Explizite Beschreibung aller benötigten Konstruktionsdaten durch ein DB-Schema
- Entwurf und Implementierung einer objektunterstützenden DB-Schnittstelle
- Anwendung von Software-Engineering-Prinzipien bei der Entwicklung DB-gestützter Ingenieursysteme
- Möglichst genaue und vollständige Verkörperung der Ingenieurobjekte an der Anwendermodell-Schnittstelle
- Allgemeine Leistungsanalyse und Bestimmung des durch die DB-Verwaltung verursachten Mehraufwandes.

KUNICAD ist kein Produktionssystem für den industriellen Einsatz, da aus Gründen der Aufwandsbeschränkung eine Reihe von Vereinfachungen bei der Modellierung in Kauf genommen wurde. Außerdem wurde nicht der volle Funktionsumfang eines geometrischen Modellierungssystems angestrebt. Das entwickelte System läßt sich vielmehr als Prototyp einer CAD-Anwendung auffassen, bei der besonderes Gewicht auf die Bereitstellung komplexer Datenverwaltungsfunktionen gelegt wurde. Seine Analyse soll wichtige Hinweise für den Entwurf geeigneterer Datenmodelle und Operationen liefern. Um nämlich angepaßte Konzepte, Architekturen und Schnittstellen von DBS für Non-Standard-Anwendungen (NDBS) entwickeln zu können, muß genügend Wissen über die beabsichtigten Anwendun-

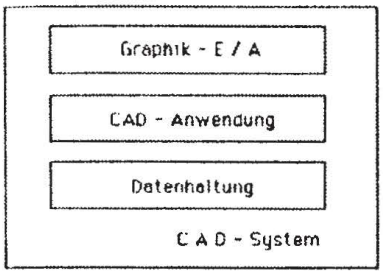


Abb. 1. Grobarchitektur eines CAD-Systems

gen, deren Typen, Verteilungen und Mengengerüste sowie deren Aufsuch- und Aktualisierungshäufigkeiten bekannt sein. Die Akquisition dieses Wissens für CAD-Anwendungen ist in [17] beschrieben.

2. Grobarchitektur des CAD-Systems und Modellierungskonzepte

Die Grobarchitektur eines CAD-Systems, wie sie in allen existierenden Systemen wiederzufinden ist, wurde in Abb. 1 skizziert, wobei die drei wesentlichen Komponenten und ihre hierarchischen Abhängigkeiten veranschaulicht sind. In unserem Fall stellt der Bauteilmodellierer die CAD-Anwendung dar. Zum Benutzer hin benötigen wir eine leistungsfähige, interaktive Graphikkomponente; sie wurde als geräte- und anwendungsunabhängiges Modul konzipiert, wobei auf vorhandene Graphikgrundsoftware zurückgegriffen werden konnte. Durch intensive Normierungsbemühungen in den letzten Jahren sind ihre Funktionen zur Gestaltung der Benutzerschnittstelle weitgehend festgelegt [1, 8, 11].

In einer CAD-Anwendung fallen vielfältigste Aufgaben der Datenverwaltung an, die sich durch komplexe Strukturen und großen Umfang der Daten sowie durch ihre durch Konstruktion und Modifikation bedingten Änderungen ergeben. Bei kommerziellen Anwendungen hat sich das Herauslösen aller Aufgaben der Datenverwaltung und ihre Systematisierung und Standardisierung in DBS seit über einem Jahrzehnt bewährt. Obwohl die Datenstrukturierung und -verwaltung erheblich höhere Anforderungen an die Datenhaltung stellt [6, 16], scheint eine solche Verfahrensweise auch bei CAD-Anwendungen angemessen. Voraussetzung für einen erfolgreichen DBS-Einsatz ist die Bereitstellung einer geeigneten „hinreichend abstrakten" Schnittstelle für die CAD-Abbildung. In unserem System ist also der 3D-Modellierer „optimal" zu unterstützen, d.h. die Komponente, die die Transformation von der „Bauteilsicht" des Benutzers in die rechnerinterne „Datensicht" zu leisten hat. Die Art dieser Unterstützung kann erst konkretisiert werden, wenn mehr über Aufgaben und Strukturen der Modellabbildung bekannt ist.

CAD-Anwendungen lassen sich am besten über die von ihnen benötigten Informationen charakterisieren, da diese sehr deutlich die Struktur- und Darstellungsvielfalt der Entwurfsobjekte und die Komplexität des Konstruktionsvorgangs widerspiegeln. Beim CAD im Maschinenbau - der körperorientierten Konstruktion - werden diese Informationen durch das *Produktmodell* oder Werkstückmodell zusammengefaßt. Wegen ihrer Vielfalt und Unterschiedlichkeit sind diese Informationen oft folgendermaßen untergliedert [7]:

Geometrisches Modell als Darstellung der geometrischen und topologischen Körperdaten sowie der bemaßungs- und darstellungstechnischen Angaben.

Produktstrukturmodell als Beschreibung von Struktur- und Zusammensetzinformationen (konstruktive Information) von Baugruppen und -teilen.

Technologisches Modell als Zusammenfassung von Angaben über Werkzeuge, Arbeitsgänge, Bearbeitungstoleranzen und Behandlung von Werkstoffen usw.

Physikalisches Modell als Informationssammlung über Stoffeigenschaften, Fertigkeiten, Gewicht, Wärmedehnung usw.

Neben den Produktdaten werden während der Konstruktion noch eine Reihe von *organisatorischen Daten* geführt, die den Zustand der einzelnen Konstruktionsvorgänge und den des Gesamtsystems festhalten. Diese verschiedenen Beschreibungsformen setzen sich als einzelne Aspekte additiv zu einer Gesamtbeschreibung des Objektes zusammen. Zur genauen Darstellung des Konstruktionsablaufes werden oft noch weitergehende Anforderungen erhoben, die das Halten von Objektversionen und Entwurfsalternativen betreffen [21]. Sie führen neben der Speicherungskomplexität erhebliche Probleme der Konsistenzkontrolle ein und sind deshalb in unserem Ansatz nicht berücksichtigt.

Die Kernprobleme beim Entwurf eines 3D-Modellierers liegen in der körperlichen Repräsentation der Werkstücke und in der Ausführung der bei ihrer Konstruktion benötigten Operationen. Deshalb werden wir nur die hierbei relevanten Aspekte, insbesondere die Modellierungsverfahren diskutieren.

Es gibt keine einheitliche Auffassung darüber, in welcher Weise der Konstrukteur bei der Konstruktion von Werkstücken vorangeht, d.h., wie der Konstruktionsprozeß am besten unterstützt werden sollte. Simuliert man mit dem Rechner bloß die Arbeit am Zeichenbrett, so verzichtet man auf viele Darstel-

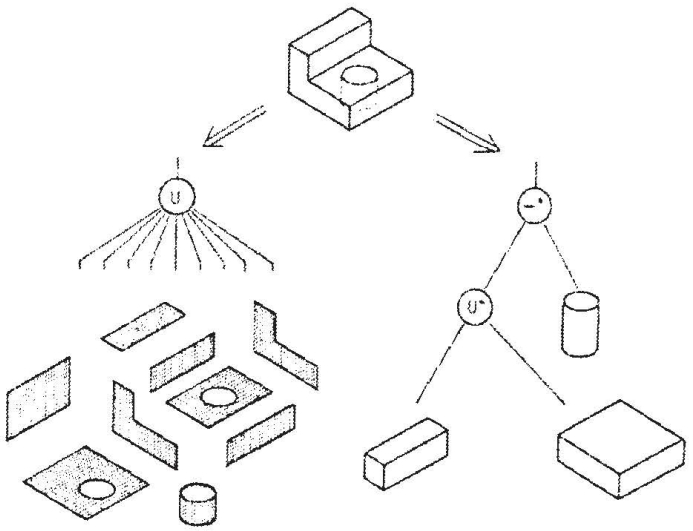


Abb. 2. Konstruktionsprinzip bei verschiedenen Modellierungsverfahren, a flächenorientierte Verfahren, b körperorientierte Verfahren

lungshilfen und Entwurfskontrollen, die durch mehrdimensionale graphische Repräsentation und Rechnereinsatz verfügbar sind. Solche zeichnungsorientierten Verfahren verlangen die Darstellung eines technischen Objektes durch verschiedene Ansichten (Risse). Seine räumliche Darstellung - beispielsweise zur Entwurfskontrolle - hat durch Rekonstruktion der 3D-Information aus den einzelnen Ansichten zu erfolgen [9]. Diese Rekonstruktion muß mühsam im Dialog erfolgen, da bei den einzelnen Projektionen die räumlichen Informationen über gleiche Berandungen oder Leitlinien verlorengegangen ist. Sehr aufwendig ist dieses Verfahren jedoch bei nachträglichen Änderungen der Geometrie, da sie in allen Ansichten getrennt vorgenommen werden müssen. Ihre Konsistenzprüfung hat wiederum manuell zu erfolgen.

Aus diesen Gründen steht der körperorientierte Ansatz bei leistungsfähigen CAD-Systemen außer Frage. Ist das rechnerinterne Abbild eines Werkstücks einmal generiert, so können durch Projektionen auf seinen räumlichen Geometriedaten alle gewünschten Ansichten berechnet und graphisch dargestellt werden. Da nachträgliche Änderungen direkt in das rechnerinterne Werkstückmodell eingehen, werden sie bei der Neuberechnung von Ansichten automatisch berücksichtigt.

Da Werkstücke sehr komplexe Gebilde darstellen, ist auch ihre Konstruktion, d.h. die Modellierung ihrer geometrischen Informationen und die Erzeugung ihrer rechnerinternen Darstellung, sehr komplex. In der Praxis haben sich deshalb verschiedene Entwurfsphilosophien zur Gestaltung von Modellierungsverfahren herausgebildet. In [29] werden die einzelnen Verfahren detailliert beschrieben. Bei *linienorientierten Verfahren* werden Objekte ausschließlich durch Linien und Punkte eingegeben und repräsentiert. Da die daraus resultierende Körperabbildung oft nicht eindeutig ist und auch keine Konsistenzprüfung zuläßt, haben sie nur geringe praktische Bedeutung.

Flächenorientierte Modellierungssysteme (Boundary Representation, BREP) benutzen zur Beschreibung von Objekten deren begrenzende Flächen (Abb. 2a). Die Speicherung geschieht entweder in Form von gekrümmten oder ebenen Flächen. Bei diesen Systemen ist immer eine eindeutige Darstellung erreichbar. Die Konsistenzprüfung ist zwar möglich, erfordert jedoch einen beträchtlichen algorithmischen Aufwand. Ein großer Nachteil ist, daß sich die Eingabe komplexer Strukturen als umständlich erweist.

Körperorientierte Verfahren (Constructive Solid Geometry, CSG) basieren auf der Idee, vordefinierte Basiskörper durch Anwendung von Operationen in immer komplexere Objekte zu überführen. Die Basiskörper sind per Definition konsistent. Die verwendeten Operationen überführen sie immer in konsistente Ergebniskörper (reguläre Operatoren). Abbildung 2b veranschaulicht die Konstruktionsgeschichte eines Objektes in der Form eines CSG-Baumes. Somit stellt die Konsistenzerhaltung kein Problem dar. Die Eindeutigkeit der Darstellung ist für körperorientierte Systeme ebenfalls kein Problem. Die Eingabe von Körpern gestaltet sich verglichen mit linien- und flächenorientierten Systemen, durch die Verwendung von Basiskörpern wesentlich einfacher. Bei diesem Verfahren werden die geometrischen Daten als Baumstruktur aufgebaut. Diese Repräsentationsform hat vordefinierte Basiskörper als Blätter eines Strukturbaumes. Die inneren Knoten enthalten zugelassene Operationen (Vereinigung, Schnitt, Differenz, Rotation, Skalierung und Translation).

Zur geometrischen Modellierung wurden in den letzten Jahren zahlreiche Darstellungsschemata vorgeschlagen [2, 10]; in existierenden Modellierungssystemen werden jedoch hauptsächlich nur CSG- und BREP-Verfahren eingesetzt [12, 24], was sicher als Hinweis für ihre praktische Tauglichkeit zu werten ist. Sie lassen sich hinsichtlich ihrer Systemarchitektur in zwei Klassen einteilen [29]. Neben den „single representation"-Systemen mit ausschließlicher BREP-Darstellung gibt es „dual representation"-Systeme, die zusätzlich eine CSG-Struktur verwalten und auf die Konsistenz beider Darstellungsformen achten.

Modellierungssysteme sind oft dadurch gekennzeichnet, daß die externe Datenhaltung sehr einfach ausgelegt ist, die Modellierungsverfahren sich aus-

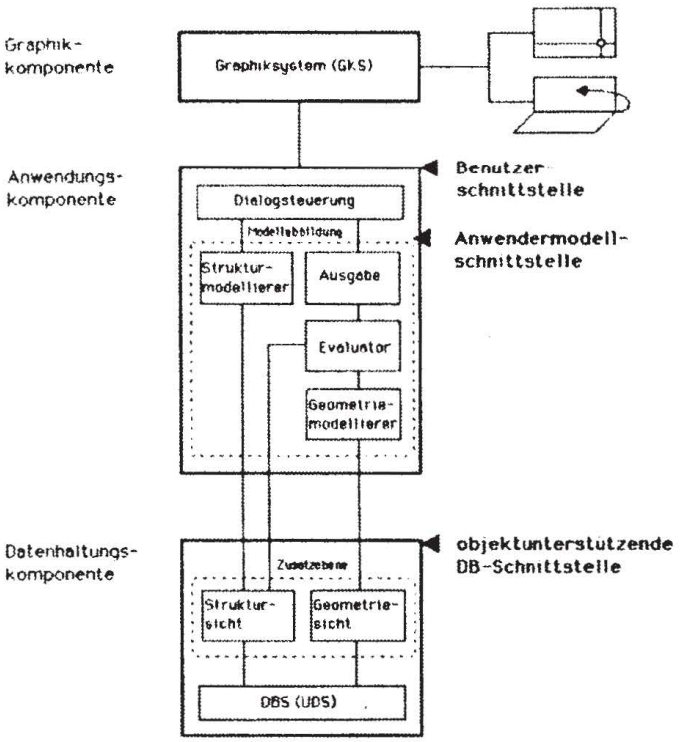


Abb. 3. Systemarchitektur

schließlich auf speziell entworfene Hauptspeicher-Datenstrukturen abstützen und die Anwendungsschnittstelle nur die typischen Operationen der geometrischen Modellierung aufweist. Unser Ansatz möchte die praxistauglichen Modellierungsverfahren datenbankgestützt einsetzen, was die Berücksichtigung von Externspeichereinflüssen, Logging-Aufwand für die Recovery usw. impliziert. Durch die Abbildung auf eine DB-Schnittstelle sind die Konstruktionsobjekte allgemein nutzbar; sie müssen deshalb explizit, d.h. unabhängig vom CAD-Programm, dargestellt und verwaltet werden. Erfahrung mit datenbankgestützter Modellierung gibt es bisher kaum; als wichtige Referenz kann [9] gelten, wo ein System für die 2D-Konstruktion beschrieben wird. Die 3D-Modellierung - in unserem Fall mit einem „dual representation"-System - ist auch deshalb interessant, weil hier kontrolliert eingesetzte Redundanz für die Benutzbarkeit und Leistung des Systems erforderlich erscheint - das betrifft das automatische Nachführen der „redundanten" BREP-Darstellung als auch redundant modellierte Beziehungen in der geometrischen Darstellung, um einfachere Modellierungsalgorithmen heranziehen zu können. Neben der Realisierungsfrage gilt unser Hauptaugenmerk der Abschätzung der redundanzbedingten Verwaltungskosten. Die geometrische Modellierung erfährt somit in unserem Ansatz eine Reihe neuer unterstützender Maßnahmen auf der Seite der Datenhaltung. Aber auch die Seite des Anwendermodells wurde durch neue Ideen und Konzepte - insbesondere durch Typisierung, Instanziierung und Parametrisierung der Anwendungsobjekte - ausgestaltet.

3. Aufbau und Funktionen des KUNICAD-Systems

3.1 Systemarchitektur

In diesem Kapitel wollen wir nun die konkrete Architektur unseres datenbankbasierten 3D-Bauteilmodellierers vorstellen. Zu den Hauptkomponenten gehört zum einen die Graphikkomponente, die Hilfsmittel zur Realisierung von ergonomischen Benutzerschnittstellen zur Verfügung stellt. Ein anderer Teil des Systems, die Datenhaltungskomponente, verwaltet die rechnerinterne Darstellung der Konstruktionsdaten. Die dritte Schicht bildet die „Anwendersicht" auf die rechnerinterne „Datensicht" ab (vgl. Abb. 1). Allgemeine Überlegungen zur Architektur von NDBS-Anwendungssystemen finden sich in [16, 25]. Der Aufbau des KUNICAD-Systems entspricht der dort vorgestellten „Zusatzebenenarchitektur". Aufbauend auf einem existierenden Datenbanksystem wird eine Programmschicht realisiert, die an ihrer Schnittstelle zur Anwendungskomponente Operationen auf anwendungsbezogenen Objekten bereitstellt. Diese Programmschicht ist ein Teil der Datenhaltungskomponente und stellt eine Zusatzebene auf dem konventionellen DBS-Kern (hier: UDS [34]) dar. Die Architektur des KUNICAD-Systems wird in Abb. 3 dargestellt.

Die *Graphikkomponente* (GMS, graphic manipulation system) erlaubt der *Anwendungskomponente* die Ein- und Ausgabe graphischer und textueller Information. Die Funktionalität der Graphikschnittstelle (GML, graphic manipulation language) entspricht der von GKS Level 2b [32]. Der *Dialogsteuerungsmodul* ist Bestandteil der Anwendungskomponente. Seine Hauptaufgabe besteht in der Vorverarbeitung der Ein- und Ausgabe sowie in der Entkopplung der implementierten *Benutzerschnittstelle* von der darunterliegenden *Anwendermodellschnittstelle* (auf der Ebene des Anwendermodelles spielt es keine Rolle, ob die Benutzerschnittstelle menü- oder kommandogesteuert arbeitet bzw., ob eine Selektion durch eine „Pick"-Eingabe möglich ist oder nicht). Die weiteren Moduln der Anwendungskomponente dienen der *Modellabbildung*, d.h. der Projektion der Anwendermodell-Objekte auf die Datenstrukturen und die damit assoziierten Operationen einer im System darunterliegenden Schnittstelle (Geometrie-/Struktursicht). Der Modul *Strukturmodellierer* verwaltet und aktualisiert die organisatorischen und strukturellen Konstruktionsdaten. Hierzu verwendet

er Funktionen des Moduls *Struktursicht* an der Schnittstelle zur Datenhaltungskomponente. Der *Ausgabemodul* übernimmt vorwiegend die Transformation der gespeicherten 3D-Information in eine 2D-Darstellung, die unmittelbar auf einem Graphik-Terminal sichtbar gemacht werden kann. In dem Modul *Geometriemodellierer* wird die eigentliche Verarbeitung der Geometriedaten durchgeführt. Geometrische Körperbeschreibungen können durch reguläre Mengenoperationen zu komplexeren Körperbeschreibungen aggregiert werden. Die Datenhaltungskomponente stellt im Modul *Geometriesicht* die hierzu erforderlichen Grundfunktionen bereit. Der *Evaluator* schließlich repräsentiert das Bindeglied zwischen der strukturellen (CSG) und der geometrischen (BREP) Interndarstellung eines Werkstücks und zeichnet deshalb auch verantwortlich für die Konsistenzerhaltung zwischen Geometrie- und Strukturdaten. Dieser Programmbaustein steuert die Berechnung der BREP-Darstellung eines Körpers aus seiner CSG-Repräsentation. Hiermit ist es möglich, vom Aufbau des CSG-Baumes ausgehend, erst zu einem späteren Zeitpunkt, zu dem eine geometrische Beschreibung tatsächlich benötigt wird, diese auch herzuleiten. Die Moduln *Struktur-* und *Geometriesicht* der *Datenhaltungskomponente* stellen die Implementierung einer „Zusatzebene" auf dem zugrundeliegenden DBS dar. Sie stellen Objekte und Operationen bereit, die von den Moduln der Modellabbildung auf vorteilhafte Art und Weise zu nutzen sind. Diese Datenhaltungsschnittstelle wird mittels der Funktionen von Geometrie- und Struktursicht realisiert, die allesamt auf der CODASYL-Schnittstelle des verwendeten DBS aufbauen. Das heißt zum einen, daß jedes Datenobjekt an dieser Schnittstelle aus den konstituierenden Schemadaten (siehe Kapitel 3.3 und Abb. 5) aufgebaut werden muß, und zum anderen, daß jede angebotene Operation durch die navigierende und positionsorientierte Datenmanipulationssprache ausgedrückt werden muß.

3.2 Benutzerschnittstelle und zugrundliegendes Anwendermodell

Nachdem nun einleitend die Architektur unseres CAD-Systems vorgestellt wurde, soll im weiteren das dem System zugrundeliegende Anwendermodell näher erläutert werden. Die damit assoziierte Anwendermodellschnittstelle wird von den Moduln der Modellabbildung bereitgestellt.

Die Dialogsteuerung implementiert mit Hilfe der Graphikkomponente die Benutzerschnittstelle des Systems, die letztendlich nur eine syntaktische Aufbereitung der Anwendermodellschnittstelle darstellt. In Tabelle 1 sind die Funktionen und Kommandos dieser Benutzerschnittstelle tabellarisch aufgeführt.

Wie in Kapitel 2 schon erwähnt, umfassen die unmittelbaren Systemaufgaben die körperliche Repräsentation der Werkstücke sowie die Ausführung der zugehörigen Konstruktionsoperationen. Dabei wird eine Einschränkung auf das Geometrie- und auf das Produktstrukturmodell vorgenommen. Die Integration von Physikalischem und Technologischem Modell würde obige primäre Problemstellung verwischen und zusätzlich zur Steigerung der ohnehin ausreichend vorhandenen Systemkomplexität

Tabelle 1. Funktionen und Kommandos der Benutzerschnittstelle

Graphikoperationen

CLR	CLeaR loscht den Inhalt des Graphic-Windows
PLT	PLoT gibt den Inhalt des Graphic-Windows auf Plotter aus
SCS	Set Character Size [für den Text auf dem Bildschirm]
SGW	Set Graphic Window
SPS	Set Plot Size
SVP	Set View Point für die Projektionsberechnung

Operationen einer Bearbeitungsphase

BOS	Begin Of Session
EOS	End Of Session
ROS	Reset Of Session

Operationen einer Konstruktionsumgebung

BOC	Begin Of Construction
EOC	End Of Construction
LCE	List Construction Environment
SCE	Show Construction Environment

Operationen auf Typen

DET	DElete Type
DGT	DElete Global Type
LGT	List Global Types
LLT	List Local Types
MAT	MAke Type zur Durchfuhrung der Typisierung
RET	REname Type
SGT	Show Global Type

Operationen auf Ausprägungen

BUI	BUILt Instance zur Durchführung der Zusammensetzung
CHA	CHAnge entspricht der Kommandofolge DCO-CON
CON	CONnect erweitert eine Baugruppe
CRI	CReate Instance zur Durchfuhrung der Instanziierung
DCO	DisCOnnect reduziert eine Baugruppe
DEI	DElete Instance.
LLI	List Local Instance
MOD	MODify gleichbedeutend mit SEL jedoch hier zusätzlich noch die Möglichkeit die gewählte Komponente zu ändern
MOI	MOve Instance
OPI	OPerate Instance zur Durchführung von Durchschnitt, Vereinigung oder Differenz
REI	REname Instance
ROI	ROtate Instance
SEL	SELect zum Auswählen einer Komponente einer Baugruppe
SHI	SHow Instance

sonstige Operationen

DES	DEScribe erlaubt die Beschreibung des Konstruktionsobjektes
END	END beendet die Terminalsitzung
PARAM	define PARAMeter definiert formale Parameter
VAR	define VARiable definiert Variablen die Schreibaufwand einsparen helfen

Körper ::=
Baugruppe | Zusammensetzung (Baugruppe., Baugruppe)
Baugruppe ::=
Zusammensetzung (Baugruppe., Baugruppe) |
Translation (Baugruppe) |
Rotation (Baugruppe) |
Skalierung (Baugruppe) |
Bauteil
Bauteil ::=
Vereinigung (Bauteil, Bauteil) |
Differnz (Bauteil, Bauteil) |
Durchschnitt (Bauteil, Bauteil) |
Translation (Bauteil) |
Rotation (Bauteil) |
Skalierung (Bauteil) |
Basiskörper
Basiskörper ::=
Quader | Polyzyl

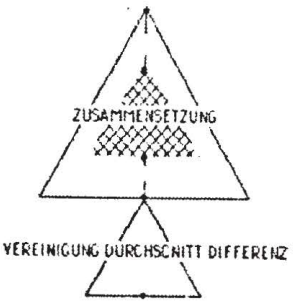


Abb. 4. Struktur der modellierten Körper

führen. Aus diesem Grunde beschreiben wir im Anwendermodell nur geometrische, konstruktive und organisatorische Werkstückinformationen. Die Sicht des Benutzers/Anwenders auf die Konstruktionsdaten ist dem körperorientierten Modellierungskonzept (siehe CSG-Modell in Kapitel 2) sehr stark angelehnt. Es werden volumenorientierte Verfahren zur Definition und Manipulation von Körperrepräsentationen angeboten. Aus Aufwandsgründen muß jedoch eine Beschränkung auf polyederförmige Körper durchgeführt werden.

Das Anwendermodell (siehe hierzu Abb. 4) unterscheidet drei Körperarten: den Basiskörper, den Bauteilkörper und den Baugruppenkörper. Die *Basiskörper* entsprechen den Grundbausteinen des CSG-Modelles, d.h., sie stellen die Primitiven dar, aus denen komplexere Körper konstruiert werden können. In unserem Modell sind dies u.a. Quader und zylinderförmige n+2-flächige Polyeder („Polyzyl"). Die möglichen Körperaggregationen werden durch die binären Modelloperatoren *Vereinigung, Durchschnitt* und *Differenz* sowie durch die unären Operatoren *Translation, Rotation* und *Skalierung* bestimmt. Zusätzlich unterstützt unser Modell noch einen Operator *Zusammensetzung*. Dies ist ein Spezialfall der Vereinigung, bei dem sich die Operanden nicht überlappen dürfen. Unter einem *Bauteilkörper* versteht man eine Verknüpfung aus Basis- bzw. wiederum Bauteilkörpern mittels der Vereinigung, dem Durchschnitt und der Differenz. Hingegen sind *Baugruppenkörper* definiert als reine Zusammensetzung von Basis-, Bauteil- bzw. Baugruppenkörpern. Sie lassen sich in ihre Bestandteile zerlegen und beschreiben damit mehr den Aufbau eines Körpers. Dagegen repräsentiert der Bereich der Bauteile die Konstruktionsgeschichte von Körpern.

Das Anwendermodell klassifiziert nicht nur Körperarten, sondern unterscheidet auch zwischen der Repräsentation eines einzelnen Körpers und der einer Körperklasse. Man spricht in diesem Zusammenhang auch von *Ausprägung* und *Typ*. Von allen drei Körperarten können sowohl Ausprägungen als auch Typen existieren. Mit Hilfe des *Instanziierungsoperators* lassen sich Ausprägungen als Elemente einer durch einen Typ bestimmten Körperklasse generieren. Umgekehrt ist auch eine *Typisierung* von Körperausprägungen möglich. Dabei wird die Strukturbeschreibung der Ausprägung zur Definition der Körperklasse verwendet. Die Typisierung stellt damit das Hilfsmittel zur Einführung neuer, komplexer Typen dar. Der weiteren Flexibilisierung des Konstruktionsprozesses dient die *Parametrisierung*. Sowohl für Typen als auch für Ausprägungen können Parameter definiert werden.

Ein weiteres, wichtiges Konzept des Anwendermodelles ist die *Konstruktionsumgebung*. Sie stellt den organisatorischen Rahmen dar, in dem Körperausprägungen und -typen entwickelt werden können. Als zugehörige Operatoren sind definiert *Beginnen, Beenden, Öffnen* und *Schließen*. Das Öffnen und Schließen einer Konstruktionsumgebung bestimmt Bearbeitungsphasen, innerhalb derer ein vom Benutzer veranlaßtes Zurücksetzen möglich ist. Dagegen wird bei Systemzusammenbruch ein Überleben der zuletzt erfolgreich durchgeführten Modelloperation garantiert. Bei Beendigung der Konstruktion wird vom System eine Globalisierung durchgeführt. Dabei werden lokale Körpertypen global bekannt gemacht, wohingegen lokale Körperausprägungen ihre Bedeutung verlieren und gelöscht werden. Diese Konstruktionsvorgänge sind DB-seitig nachzubilden, was eine Anpassung des Transaktionskonzeptes impliziert. Beispielsweise wird in [20] unterschieden zwischen langen Entwurfstransaktionen und darin enthaltenen kurzen Recovery-Transaktionen. Zusammenfassend werden alle Konzepte des Anwendermodells, in Form eines Sitzungsprotokolls, dargestellt in Tabelle 2, nochmals beispielhaft

Tabelle 2. Beispiel eines Sitzungsprotokolls

LGT

List Global Type: die Namen aller dem System bekannten globalen Typen werden aufgelistet.

Basiskörpertyp Keil , 3 formale Parameter
Basiskörpertyp Quader , 3 formale Parameter
Basiskörpertyp Spat , 4 formale Parameter
Basiskörpertyp Pyramide , 3 formale Parameter

SGT (Keil)

Show Global Type: wir fordern weitere Informationen über den Basistyp Keil an.

Basiskörpertyp Keil , 3 formale Parameter
0 Instanziierungen
Typbeschreibung:

Der Basiskörpertyp Keil besitzt drei formale Parameter: Breite, Tiefe, Höhe. Breite und Tiefe bestimmen die Grundfläche in der xy-Ebene, Höhe die Keilausdehnung in z-Richtung. Die Keilspitze liegt in der yz-Ebene. Die Generierung erfolgt durch "Keil(Höhe,Breite, Tiefe)".

BOC (Beispiel)

Begin Of Construction: Wir definieren eine neue Konstruktionsumgebung und geben dieser den Namen **Beispiel.**

Construction Environment "Beispiel" is created.

PARAM (a, b) (20, 30)

CRI (Pyramide (a, a, b); K1 (a, b))

Hiermit definieren wir eine parametrisierte Ausprägung K1 des Basiskörpertyps **Pyramide.** Die Parameterzuordnung wird durch die beiden Parameterlisten bestimmt. Die "beispielhafte" Ausprägung von K1 ergibt sich aus Pyramide (20, 20, 30).

CRI (Quader(30, 30, 30); K2)

MVI (K2 ; (0, 20, 0))

MoVe Instance: Wir verschieben die Ausprägung K2 um 20 Einheiten in Y-Richtung.

OPI (Differenz, K1, K2, T1)

OPerate Instance: Wir bilden den Differenzkörper (K1-K2) und benennen ihn T1. K1 und K2 verlieren ihren Namen und ihre Identität. Sie gehen vollständig in T1 über.

MAT (T1)

MAke Type: Durch Typisierung wird aus der Ausprägung T1 der Typ Typ T1.

CRI (T1() ; K1)

CRI (T1() ; K2)

ROI (K2, (180, 0, 0))

Wir legen vom Typ T1 die Ausprägungen K1 und K2 an. K2 wird mit "ROtate Instance" um 180 Grad um die x-Achse gedreht.

BUI ((K1, K2), K)

K1 und K2 werden zu der Baugruppe K "verklebt". Damit verlieren K1,K2 ihre Namen, nicht aber ihre Identität.

EOS

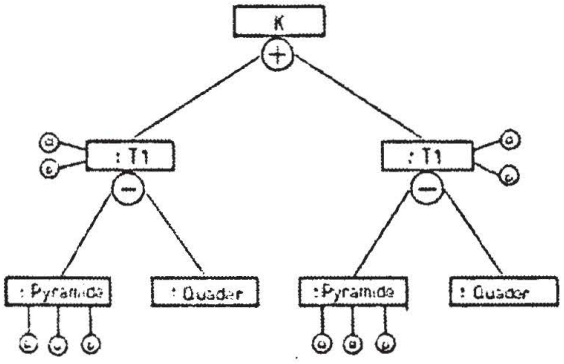
Der Konstruktionsrahmen wird geschlossen. Die Bearbeitungsphase ist beendet. Der Typ T1 und die Ausprägung K bleiben als lokale Größen bestehen.

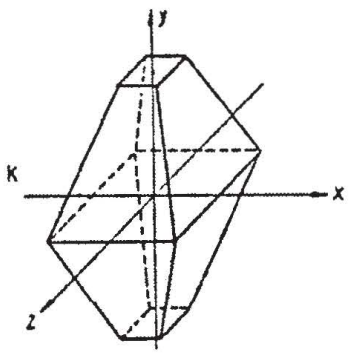
EOC

Beenden der Konstruktionsumgebung "Beispiel". Damit werden auch alle definierten Parameterzuordnungen und lokalen Ausprägungen (hier K) gelöscht. Die lokalen Typen (hier T1) werden globalisiert.

END

Beenden der Terminalsitzung.





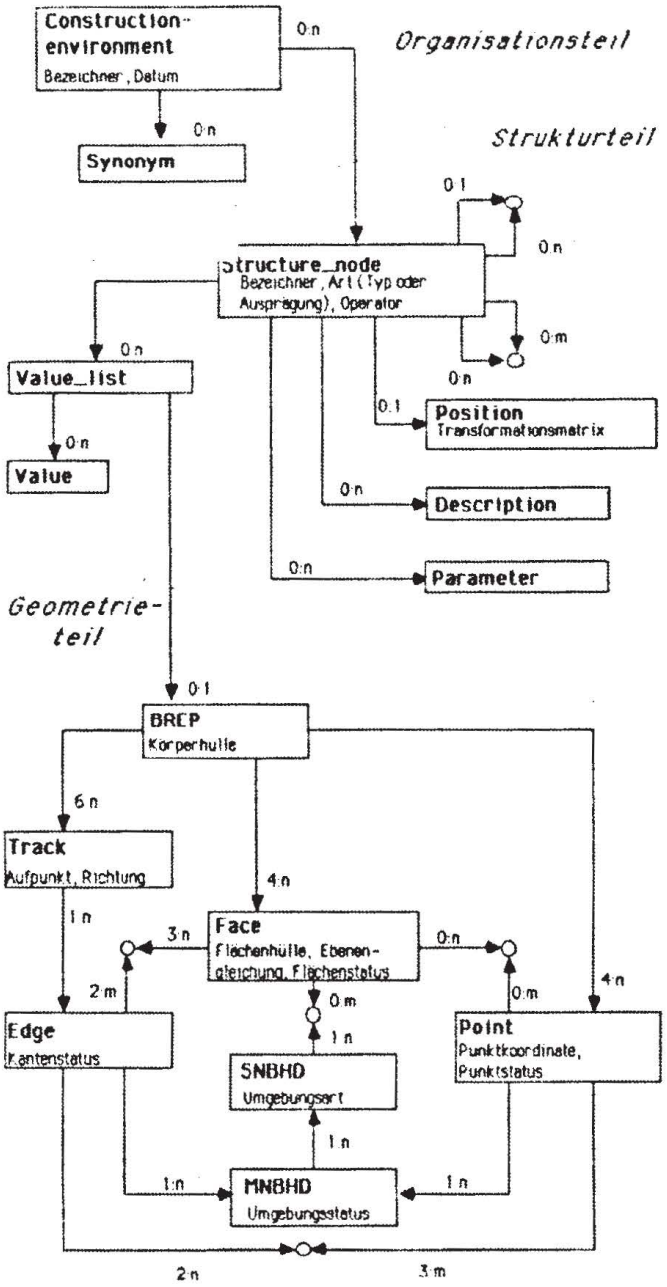


Abb. 5. Datenbankschema der CAD-Anwendung

aufgezeigt. Dabei wird direkt Bezug auf Tabelle 1 genommen.

3.3 Datenbankschema der CAD-Anwendung

Die durch das Anwendermodell beschriebene Information muß mittels des vom DBS [34] zur Verfügung gestellten netzwerkorientierten Datenmodells dargestellt werden. Obwohl erste rechnergestützte Hilfsmittel zum automatischen und systematischen Datenbankentwurf bereits existieren, scheinen diese eher für konventionelle Anwendungen geeignet. In den „nicht-kommerziellen" Bereichen hingegen ist man weiterhin auf die eigene Entwurfserfahrung angewiesen. Deshalb reflektiert das hier vorgestellte Datenbankschema unsere eigene Modellierungserfahrung, die durch Hinweise und Diskussionen mit Fachleuten [9, 24] verstärkt wurde. Eine kritische Betrachtung des konkreten Schemaentwurfs und dessen Auswirkungen auf die Leistungsfähigkeit des Gesamtsystems bleibt dem Kapitel 4 vorbehalten.

Im Datenbankschema werden drei Arten von Anwendungsinformation unterschieden, nämlich organisatorische, strukturelle und geometrische Information. In Abb. 5 ist das Diagramm des vollständigen Netzwerk-Schemas enthalten. Zusätzlich zum Namen des Satztyps werden gegebenenfalls noch die wichtigsten charakterisierenden Attribute angegeben. Die Pfeile in Abb. 5 repräsentieren (1:n)-Beziehungstypen. Der kleine Kringel steht dabei stellvertretend für den sog. Relation-Satztyp, welcher zur Modellierung der (m:n)-Beziehungen benötigt wird. Die Namen dieser Satztypen wurden aus Gründen der Übersichtlichkeit ebenso weggelassen, wie die der Beziehungstypen. Die zu den Beziehungstypen gehörenden Zahlenwerte stellen die Kardinalitätsrestriktionen der Beziehungen dar.

Der *Organisationsteil* beschreibt den Zustand der einzelnen Konstruktionsumgebungen. Für jede definierte Umgebung existiert eine Satzausprägung vom Typ Construction-environment, dem eine Menge von Synonym-Sätzen zugeordnet ist, die die Variablen- und Parameterinformation enthalten. Der (1:n)-Beziehungstyp zum Structure_node-Satztyp verknüpft die Konstruktionsumgebung mit der rechnerinternen Darstellung der in ihr definierten Typen und Ausprägungen.

Der *Strukturteil* realisiert ein Netz von Knoten (über die reflexiven Beziehungstypen auf dem Structure-node-Satztyp), durch das die „CSG"-Struktur des zugehörigen Typ- oder Ausprägungskörpers festgelegt ist. Das Operatorattribut bestimmt, ob der Körper durch Vereinigung, Durchschnitt, Differenz oder Addition der assoziierten Körper entstanden ist. Die zugeordneten Satzausprägungen vom Typ Parameter repräsentieren die aktuellen Parameter einer Instanziierung oder Zusammensetzung. Zusätzlich sind jedem Structure_node-Satz Satzausprägungen vom Typ Description für erläuternden Text sowie vom Typ Position zugeordnet. Der Position-Satz enthält eine Transformationsmatrix, in die alle Verschiebungen, Rotationen und Skalierungen des Körpers eingearbeitet werden. Die Ausprägungen des Satztyps Value_list verbinden den Strukturteil mit dem Geometrieteil. Die Existenz der geometrischen Information eines Körpers ist optional, da sie aus der stets vorhandenen Strukturinformation leicht abgeleitet werden kann. (Dies ist Aufgabe des Moduls Evaluator, siehe Abschnitt 3.1.)

Der *Geometrieteil* benutzt zur Darstellung der geometrischen und topologischen Körperdaten das

Begrenzungsflächenmodell (siehe unter „flächenorientierter Modellierung" in Kapitel 2), in dem die Körpergeometrie durch die Körpermantelflächen charakterisiert wird. Die Flächen selbst sind durch die begrenzenden Kanten und diese wiederum durch ihre Randpunkte bestimmt. Verknüpfungsmäßig handelt es sich dabei jeweils um (m:n)-Beziehungen. Zusätzlich zu dieser redundanzfreien Modellierung bestehend aus BREP-, Face-, Edge- und Point-Satztyp sowie den zugehörigen Beziehungstypen, haben wir eine Reihe von Zusatzinformation explizit dargestellt. Diese ist zwar durch umständliche Berechnungen aus dem redundanzfreien Schema ableitbar, ihre redundante Speicherung erscheint aber gerechtfertigt, da sie zu einer erheblichen Vereinfachung der Algorithmen innerhalb der BREP-Verarbeitung führt (vgl. [30]).

Der BREP-Satztyp enthält als beschreibendes Attribut u.a. die sog. Körperhülle. Sie besteht aus den Endpunkten der Raumdiagonalen eines den eigentlichen Körper umhüllenden Quaders. Diese Information wird zur Erkennung von etwaigen Körperüberlappungen benötigt. Der Track-Satz dient der Partitionierung der Kantenmenge eines Körpers. Einer Ausprägung sind alle jeweils auf einer Geraden liegenden Kanten zugeordnet. Die Attribute des Face-Satztyps bestehen u.a. aus der Flächenhülle in Form eines umhüllenden Quaders, einer Ebenengleichung und einem Statusattribut. Die Ebenengleichung definiert zusätzlich zur assoziierten Ebene noch die Ausrichtung des Körperinneren relativ zur Fläche. Der Point-Satztyp besitzt die Punktkoordinaten, die die eigentliche geometrische Information beinhalten sowie ein Statusattribut. Die Bedeutung des Edge-Satztyps beruht allein auf den mit ihm assoziierten Satztypen. Das Attribut Kantenstatus erfüllt, wie auch der Flächen- und Punktstatus, nur einen organisatorischen Zweck während der Verknüpfungsphase zweier Begrenzungsflächendarstellungen. Der MNBHD-Satztyp (multiple neighbourhood) und der SNBHD-Satztyp (single neighbourhood) werden aufgrund des eingesetzten Verfahrens zur BREP-Verarbeitung benötigt. Der verwendete Algorithmus basiert auf den in [5, 23, 30] beschriebenen Verfahren. Beide Satztypen dienen der genaueren Beschreibung der geometrischen Umgebung eines Punktes bzw. einer Kante bezogen auf einen Körper. Diese Information ist insbesondere bei einer Körperberührung von Bedeutung. Da sich Körper quasi selbst berühren können, sind „Mehrfachumgebungen", bestehend aus mehreren „Einfachumgebungen" denkbar. Daher bewirkt der MNBHD-Satz lediglich eine Gruppierung der SNBHD-Sätze. Das Attribut Umgebungsart erlaubt Flächen-, Kanten- und Eckpunktumgebungen zu unterscheiden. Zusätzlich sind mit einem SNBHD-Satz alle die Einfachumgebung bestimmenden Flächen über einen Relation-Satz verbunden. In Abb.6 sind die verschiedenen Punktmengenumgebungen, zusammen mit den sie bestimmenden Flächen (bzw. Winkeln), für einen einfachen Quader aufgezeigt. Ein Beispiel für eine Mehrfachumgebung, die aus zwei Einfachumgebungen (nämlich einer Flächen- und einer Kantenpunktumgebung) besteht, ist in Abb.7 dargestellt. Eine MNBHD-Satzausprägung repräsentiert entweder eine Kanten- oder eine Punktumgebung. Dabei sind die (1:n)-Beziehungen zwischen Edge- und MNBHD-Satz bzw. zwischen Point- und MNBHD-Satz Alternativen zueinander. Ebenfalls eine besondere Bedeutung besitzt der (m:n)-Beziehungstyp zwischen dem Face- und dem Point-Satz. Er wird nur temporär zur Repräsentation von sog. Durchdringungspunkten während der Verknüpfungsphase zweier Begrenzungsflächendarstellungen benutzt. Das sind Punkte, die auf einer Kante des einen und auf einer Fläche des anderen Verknüpfungskörpers liegen.

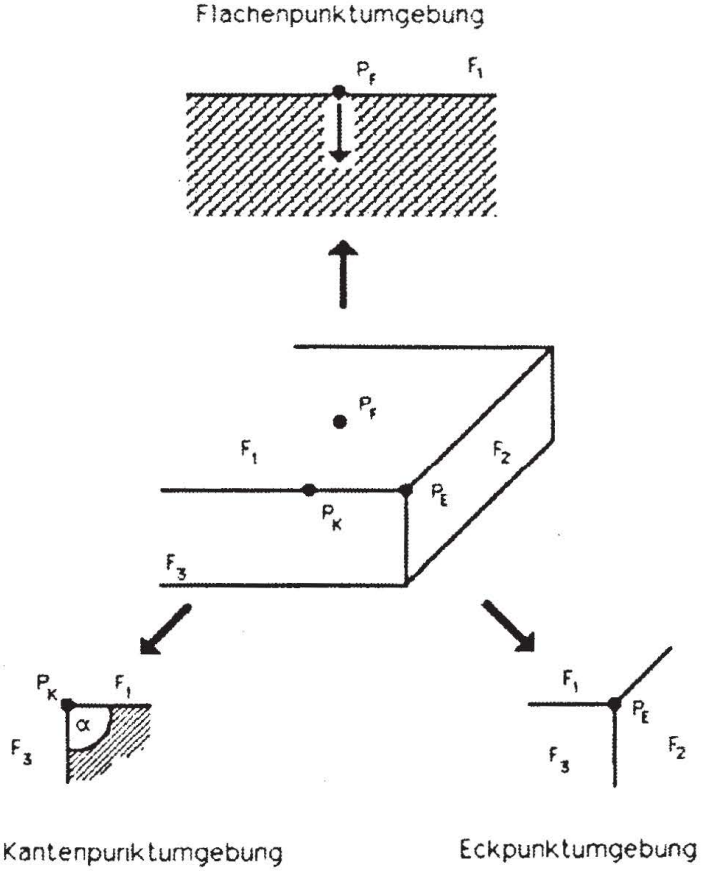


Abb.6. Die verschiedenen Punktumgebungen

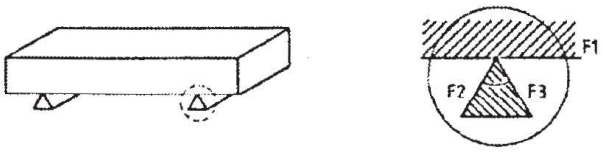


Abb.7. Beispiel einer Mehrfachumgebung

Tabelle 3. Verarbeitungseinheiten und Operationen der Geometriesicht

Verarbeitungseinheiten der Geometriesicht:

Verarbeitungs-einheit	Anker	konstituierende Satztypen
BREP	BREP	Track Face Edge Point MNBHD SNBHD
Fläche	Face	Track Edge Point MNBHD SNBHD
Kante	Edge	Point MNBHD SNBHD Face
Punkt	Point	MNBHD SNBHD Face
Umgebung	MNBHD	SNBHD Face

Operationen auf Kanten:

- GET_ATTRIBUTES_OF_EDGE: - selbsterklärend -
- GET_POINTS_OF_EDGE: - selbsterklärend -
- GET_NBHD_OF_EDGE: Die Umgebungsbeschreibung einer Kante wird aus dem mit der Kante assoziierten Umgebungsobjekt (MNBHD, SNBHD, Face) aggregiert und als Hauptspeicherstruktur übergeben.
- SET_NBHD_OF_EDGE: - selbsterklärend -
- DEFINE_EDGE_BY_POINTS_AND_EDGE: Zwei Punkte werden zu einer neuen Kante zusammengefügt. Diese erhält die Umgebung einer Bezugskante.
- DEFINE_EDGE_BY_POINTS_AND_NBHD: Die Umgebung einer neuen Kante, die durch zwei Punkte bestimmt ist, wird explizit als Argument übergeben (sonst wie DEFINE_EDGE_BY_POINTS_AND_NBHD).
- DELETE_EDGE: - selbsterklärend -
- CONNECT_EDGES: Zwei Kanten, die einen Randpunkt gemeinsam haben und auf einer Geraden liegen, werden zu einer einzigen Kante zusammengefaßt.
- MARK_EDGE: Markieren einer Kante mit einem bestimmten Verarbeitungszustand.
- ADDEDGE: Eine Kante wird mit ihren konstituierenden Punkten, als Kante einer bestimmten Fläche in eine BREP-Struktur eingebracht.

Operationen auf Punkte:

- GET_ATTRIBUTES_OF_POINT: - selbsterklärend -
- GET_NBHD_OF_POINT: Die Umgebungsbeschreibung eines Punktes wird aus dem mit dem Punkt assoziierten Umgebungsobjekt (MNBHD, SNBHD, Face) aggregiert und als Hauptspeicherstruktur übergeben.
- SET_NBHD_OF_POINT: - selbsterklärend -
- ADD_CROSSPOINT: Ein "Durchdringungspunkt", also der Schnittpunkt einer Kante mit einer Fläche wird in die BREP-Struktur eingebracht (falls er noch nicht vorhanden ist). Gleichzeitig wird eine entsprechende Umgebung aufgebaut.

Operationen auf BREP-Strukturen:

- MERGE_BREPS: Es werden zwei BREP-Körperdarstellungen "gemischt". Hierdurch wird eine Arbeitsgrundlage für weitere Modifikationen geschaffen.
- GET_ATTRIBUTES_OF_BREP: Die einfach strukturierten Attribute werden übergeben (z.B. Körperhülle in Form der Koordinaten zweier Eckpunkte).
- GET_FIRST/NEXT_FACE_OF_BREP: Die Übergabe einer Fläche erfolgt als "expliziter Cursor".
- GET_FIRST/NEXT_EDGE_OF_BREP: - selbsterklärend -
- CHECK_BREP: Die Körper- und Flächenattribute, die sich als Aggregation aus Attributwerten der Objektkonstituenten (Edge, Point) ergeben, werden aktualisiert (z.B. das Hüllenattribut des BREP- bzw. Face-Records).
- ADDBREP: Die Generierung einer neuen BREP-Struktur (Aufbau eines Basis-Körpers) wird vorbereitet. Es erfolgt die Speicherung eines "Dummy"-BREP-Records.
- DELETE_BREP: - selbsterklärend -
- UPBREP: Das Brep-Hüllenattribut wird mit den entsprechenden Face-Hüllenattributen aktualisiert.

Operationen auf Flächen:

- GET_ATTRIBUTES_OF_FACE: - selbsterklärend -
- GET_FIRST/NEXT_EDGE_OF_FACE: - selbsterklärend -
- GET_COMMON_POINTS_OF_FACES: Alle gemeinsamen Punkte, insbesondere die Durchdringungspunkte, werden aufgesucht und in einer Argumentliste übergeben.
- GET_EDGES_OF_FACES_ON_SAME_TRACK: - selbsterklärend -
- GET_FACES_ON_SAME_SURFACE: - selbsterklärend -
- DELETE_FACE: - selbsterklärend -
- CONNECT_FACE: Zwei Flächen werden zu einer zusammengefaßt, falls sie in einer gemeinsamen Ebene liegen und sich für beide das Körperinnere auf der gleichen Seite befindet.
- MARK_FACE: Markieren einer Fläche mit einem bestimmten Verarbeitungszustand.
- ADDFACE: Die Generierung einer neuen Fläche wird durch Speicherung eines Face-Records eingeleitet.
- CONFACES: Die Umgebungsinformation für die gemeinsamen Kanten zweier Flächen wird in der DB aufgebaut.

Das dargestellte DB-Schema enthält, wie z.T. schon erwähnt, eine Reihe von Redundanzen, durch die zum einen das hier gewählte Verarbeitungsverfahren unterstützt und zum anderen eine Verbesserung des Zugriffsverhaltens erreicht werden soll (siehe Kapitel 4). Neben der inhaltlichen Redundanz von Geometrie- und Strukturteil existiert innerhalb des Geometrieteils Redundanz auf unterschiedlichen Ebenen. Die Hüllenattribute sowie die Track- und Ebenengleichungen des BREP-, Face und Track-Satztyps sind jeweils aggregierbar. Zusätzlich zu dieser sog. „Attributredundanz" gibt es noch eine „strukturelle Redundanz". Beispielsweise tritt diese Art der Redundanz zum Vorschein durch die Beziehungen der Edge- und Point-Sätze zu den Umgebungsrepräsentationen und damit auch zu denjenigen Face-Sätzen, mit denen sie direkt über die Face-Edge-Beziehung bzw. indirekt über die Edge-Point-Beziehung bereits assoziiert sind. Die Erhaltung der Konsistenz dieser teilweise redundanten Information liegt vollständig in der Verantwortung derjenigen Algorithmen der DBS-Zusatzebene, die auf diesen Schemaausschnitten operieren.

3.4 Objektunterstützende DB-Schnittstelle

Die Datenhaltungskomponente besteht aus drei Teilen, dem *Datenbanksystem* (UDS), dem Modul *Geo-*

Tabelle 4. Verarbeitungseinheiten und Operationen der Struktursicht

Verarbeitungseinheiten der Struktursicht:

Verarbeitungs-einheit	Anker	konstituierende Satztypen
Konstruktions-umgebung	Construction environment	Synonym, Structure_node, (gesamtes Schema)
Synonym	Synonym	-
Instanz/Typ	Structure_node	Position, Description, Parameter, Value_list, Value, BREP,...

Operationen auf Konstruktionsumgebungen:

- GET_CONSTRUCTION_BY_NAME: - selbsterklärend -
- GET/SET_ATTRIBUTES_OF_CONSTRUCTION: "
- DELETE_CONSTRUCTION: "
- GET_SYNONYM_OF_CONSTRUCTION_BY_NAME: "
- GET_FIRST/NEXT_INSTANCE_OF_CONSTRUCTION: "
- GET_FIRST/NEXT_TYPE_OF_CONSTRUCTION: "
- GET_INSTANCE_OF_CONSTRUCTION_BY_NAME: "
- GET_TYPE_OF_CONSTRUCTION_BY_NAME: "

Operationen auf Synonymen:

- GET/SET_ATTRIBUTES_OF_SYNONYM: - selbsterklärend -

Operationen auf Instanzen/Typen:

Strukturknoten (Satztyp Structure_node im DB-Schema) repräsentieren sowohl Instanzen als auch Typen. Daher wird im folgenden immer dann, wenn sich eine Operation sowohl auf Instanzen als auch auf Typen bezieht, der Begriff NODE verwendet.

- GET_PREDECESSOR_OF_NODE: Es wird der "Typvorgänger" eines NODE über die "Ist_Instanziierung_von"-Beziehung ermittelt.
- GET_CHILD_INSTANCE_BY_INDEX: Die durch einen Index spezifizierte Komponentenausprägung wird bestimmt ("setzt_sich_zusammen_aus"-Beziehung)
- GET/SET_ATTRIBUTES_OF_NODE: - selbsterklärend -
- ATTACH_INSTANCE_TO_TYPE: "
- ATTACH_CHILD_NODES_TO_INSTANCE: "
- DELETE_TYPE: "
- DELETE_INSTANCE: "
- MAKE_TYPE: "
- GET/SET_POSITION_OF_NODE: "
- GET/SET-PARAMLIST_OF_NODE: "
- GET_FIRST/NEXT_VALUES_OF_NODE: "
- ATTACH_VALUE_TO_NODE: "
- DELETE_VALUES_OF_NODE: "
- GET_BREP_OF_NODE: "
- ATTACH_BREP_TO_NODE: "

metriesicht und dem Modul *Struktursicht*. Geometrie- und Struktursicht bilden die Zusatzebene der Datenhaltungskomponente und definieren die Schnittstelle zur Anwendungskomponente. Diese wollen wir als *objektunterstützende Schnittstelle (OSS)* bezeichnen. Hiermit soll zum Ausdruck kommen, daß der „Gegenstand der Verarbeitung" (Verarbeitungseinheit, VE) von komplexerer Natur sein kann als herkömmliche DB-Sätze. Wir wollen an dieser Stelle keine vollständige Beschreibung aller Operationen geben, sondern vielmehr anhand einiger Beispiele aus der Geometriesicht eine Charakterisierung dieser Schnittstelle durchführen (eine Zusammenstellung aller Operationen der Geometrie- bzw. Struktursicht gibt Tabelle 3 bzw. Tabelle 4).

Alle OSS-Operationen nehmen Bezug auf sog. *explizite Cursor*. Unter einem expliziten Cursor verstehen wir eine spezielle Referenz auf eine Satzausprägung (im folgenden als Ankersatz bezeichnet) und deren „unmittelbaren Nachbarausprägungen" in der Datenbank, d.h. eine Referenz auf alle mit dem Ankersatz (über Set-Beziehungen) assoziierten Satzausprägungen. Der explizite Cursor beschreibt daher die „Lage" eines Satzes innerhalb einer einstufigen Struktur von weiteren Satzausprägungen. Darüberhinaus repräsentiert er *implizit* die gesamte mit dem Ankersatz assoziierte Information. Die Realisierung des expliziten Cursor-Konzeptes erfolgt unter Ausnutzung des CODASYL-spezifischen Datenbankschlüssels (DB-Key). In einer Cursor-Struktur sind neben notwendiger Verwaltungsinformation eine Reihe von DB-Keys zusammengefaßt, durch die auf aktuell benötigte Satzausprägungen bestimmter Satztypen (implizit festgelegt durch den Typ der VE) verwiesen wird. Programme der Anwendungskomponente besitzen keinen direkten Zugriff auf den expliziten Cursor; sie können daher nicht zwischen dem Cursor und der damit assoziierten Verarbeitungseinheit unterscheiden (Abb. 8). Eine weitere Eigenheit der expliziten Cursor besteht darin, daß sie in prinzipiell beliebiger Anzahl generiert werden können. Als spezielle Cursor-Operatoren sind definiert: ACTIVATE, DEACTIVATE und ASSIGN. Der Operator ASSIGN spielt die Rolle einer Zuweisung auf Cursor-Ebene.

Die VE der objektunterstützenden Schnittstelle korrespondieren mit dem zugrundeliegenden DB-Schema. So beinhaltet die Geomtriesicht die BREP-VE, die sich auf der Schemaebene aus dem gesamten Geometrieteil zusammensetzt (Tabelle 4) auf Ausprägungsebene sind mit ihr alle diejenigen Satzausprägungen assoziiert, die mit einem einzigen BREP-Ankersatz über entsprechende Set-Beziehungen verbunden sind. Der auf BREP-VE definierte MERGE-

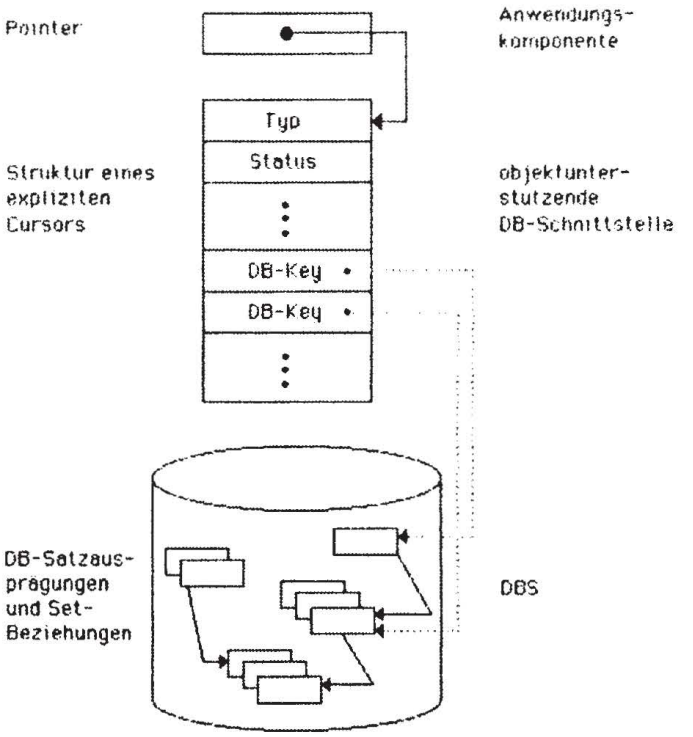


Abb. 8. Schematisierte Darstellung eines expliziten Cursor

Operator stellt die komplexeste Operation an der Schnittstelle zur Geometriesicht dar. Mit ihr wird - durch ein algorithmisch äußerst aufwendiges „Mischen" zweier BREP-Körperdarstellungen - eine Arbeitsgrundlage geschaffen, auf der die eigentliche Körperverknüpfung durch den Geometriemodellierer abgewickelt werden kann. Weitere VE sind Fläche-, Kante-, Punkt- und die Umgebung-VE. Sie bestehen aus Satzstrukturen, die im wesentlichen durch eine Reihe konstituierender Satztypen sowie einem Ankersatz bestimmt sind.

Auf diesen VE sind sowohl einfache als auch komplexe Operationen definiert. Zu den einfachen zählen u.a.: GET_ATTRIBUTES_OF_FACE, SET_ATTRIBUTES_OF_EDGE, GET_NEXT_EDGE_OF_FACE, ... Hierbei werden einfache charakterisierende Attribute bzw. Inkarnationen expliziter Cursor als Argument übergeben. In der Regel ist eine solche Operation auf wenige DB-Operationen abzubilden. Als Operationen, bei denen ein Argument, allgemeiner der „Gegenstand der Verarbeitung", eine kompliziertere Struktur aufweist, sind u.a. die folgenden zu nennen: GET_COMMON_POINT_OF_FACES, ADD_CROSSPOINT, GET_NBHD_OF_POINT, DEFINE_EDGE_BY_POINTS_AND_NBHD, ... Diese Art von Operationen lassen sich nochmals in zwei Gruppen unterteilen. Bei der ersten Gruppe handelt es sich um Operationen die im wesentlichen das „Lesen" bzw. das „Schreiben" einer Satzstruktur bewirken (z.B. GET_NBHD_OF_POINT, SET_NBHD_OF_POINT). Die zweite Gruppe führt dagegen unmittelbar eine komplexe Verarbeitung involvierter Satzstrukturen durch. Die oben erwähnte Operation ADD_CROSSPOINT gibt hierfür ein Beispiel. Durch sie wird ein Punkt als gemeinsamer Kanten- und Flächenpunkt (Durchdringungspunkt) in eine BREP-Struktur eingebracht. Als weiteres markantes Beispiel ist die Merge-Operation zu nennen.

Zusammenfassend können wir feststellen, daß an der objektunterstützenden Schnittstelle grob drei Operationstypen zu erkennen sind:

- *satzorientierte Operationen:* Der Gegenstand der Verarbeitung ist genau ein Satz.
- *strukturorientierte Operationen:* Der Gegenstand der Verarbeitung ist eine strukturierte Menge heterogener Sätze. Die Verarbeitung ist geprägt durch das „Herausziehen" solcher Satzmengen aus der bzw. ihr „Einbringen" in die DB.
- *objektorientierte Operationen:* Der Gegenstand der Verarbeitung ist ebenfalls eine strukturierte Menge heterogener Sätze. Allerdings ist die Verarbeitung durch eine „semantisch höherwertige" Aktivität (z.B. Mischen zweier BREP-Strukturen) bestimmt.

Der Übergang zwischen den Operationstypen erscheint fließend. So gibt es Operationen, die Aspekte mehrerer Typen aufweisen. Bei GET_COMMON_POINTS_OF_FACES handelt es sich einerseits um eine strukturorientierte Operation, da eine strukturierte Menge (eine Liste) von Point-Sätzen aufgebaut und als Argument übergeben wird. Andererseits muß zum Auffinden der gemeinsamen Punkte zweier Flächen ein Algorithmus durchlaufen werden, der verschiedene Teile der BREP-Struktur berührt (hierbei handelt es sich um eine „semantisch höherwertige" Aktivität). Die oben vorgestellten Operationstypen dienen demnach nicht zur vollständigen Klassifizierung, wohl aber zur Charakterisierung der Verarbeitung an der objektunterstützenden Schnittstelle.

4. Analyse des KUNICAD-Systems

Das KUNICAD-System besteht ungefähr aus 20000 Anweisungszeilen und wurde in den Programmiersprachen PL/I, FORTRAN 77 und PASCAL implementiert. An Software-Paketen werden das Datenbankverwaltungssystem UDS [34] und das Graphiksystem SIGMEX 6161 [32] benutzt. Damit ist das KUNICAD-System auf allen BS2000-Rechenanlagen der Firma Siemens mit entsprechender Graphik-Hard- und Software lauffähig.

Nachdem im vorangehenden Kapitel eine detaillierte Beschreibung unseres KUNICAD-Systems gegeben wurde, soll hier eine erste kritische Analyse

dieser Systementwicklung hinzugefügt werden. Zuerst erfolgt eine Bewertung der unmittelbaren Systemeigenschaften, die sowohl das realisierte Anwendermodell und die entwickelte Graphikkomponente als auch wichtige Systeminterna mit einschließen. Die Brauchbarkeit der objektunterstützenden Schnittstelle zur Datenhaltungskomponente wird in einem eigenen Abschnitt diskutiert. Erste Leistungsuntersuchungen am Prototyp ergänzen die Systemanalyse und leiten über zu der Bewertung der Systemarchitektur und des Verarbeitungskonzeptes.

4.1 Bewertung der Systemeigenschaften

Die adäquate Unterstützung des Konstruktionsvorganges ist die zentrale Aufgabe eines CAD-Systems. Das Anwendermodell unseres KUNICAD-Systems beinhaltet daher allgemeine konstruktionsunterstützende Konzepte. Durch die Unterscheidung von drei Körperarten (Basis-, Bauteil- und Baugruppenkörper) im Anwendermodell kann eine detailliertere Konstruktionssemantik ermöglicht werden. Die Typisierung/Instanziierung erlaubt den Übergang von der Ausprägung zum Typ bzw. umgekehrt vom Typ zur Ausprägung. Damit ist die Einführung von neuen und komplexeren Körperklassen möglich. Ebenso liefert das Konzept der Typen- bzw. Ausprägungsparametrisierung eine zusätzliche Flexibilisierung des Konstruktionsprozesses (siehe Konstruktionsbeispiel in Abschnitt 3.2). Schließlich stellt die Konstruktionsumgebung einen benutzerkontrollierten organisatorischen Rahmen zur Verfügung, in dem Körperausprägungen und -typen nach bestimmten Verarbeitungsregeln entwickelt werden können.

Aufbauend auf diesem Anwendermodell lassen sich nun ergonomische Benutzerschnittstellen realisieren, die sich dem Arbeitsablauf und den Denkgewohnheiten im CAD-Bereich anpassen und damit die Akzeptanz solcher Systeme für kreative Entwurfsarbeiten entscheidend beeinflussen. Unsere Benutzerschnittstelle basiert auf einem Graphiksystem gemäß GKS-Standard (Level 2b). Der Übergang zu einem 3D-GKS mit 3D-Eingabe wäre allerdings sehr komfortabel und wünschenswert.

Das KUNICAD-System stellt gemäß Kapitel 2 ein körperorientiertes „dual representation"-Modellierungssystem dar. Die rechnerinterne Darstellung der Konstruktionskörper entspricht einer CSG/BREP-Kombination. Die primäre, CSG-artige Darstellung erlaubt die direkte und einfache Abbildung des Anwendermodells, während sich die optionale BREP-Darstellung, wegen ihrer Nähe zur Graphik-Repräsentation, sehr gut für Ausgabezwecke eignet. Die erforderliche CSG-BREP-Transformation soll durch die im folgenden analysierte objektunterstützende Schnittstelle gezielt unterstützt werden.

4.2 Brauchbarkeit der objektunterstützenden DB-Schnittstelle

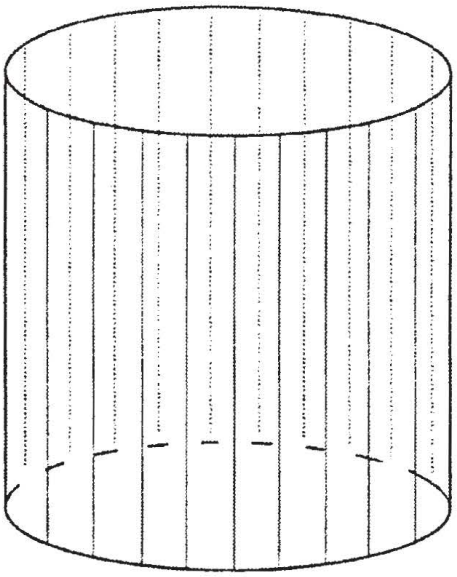
Die Datenhaltungskomponente stellt für die Anwendungskomponente eine objektunterstützende Schnittstelle bereit. Von zentraler Bedeutung ist die Unabhängigkeit dieser Schnittstelle von dem darunterliegenden DBS sowie ihre einfache anwendungsbezogene Erweiterbarkeit. Analog zur Geometrie- und Struktursicht läßt sich eine Sicht zur Unterstützung von Material- und Festigkeitsberechnungen entwickeln oder nach einer Schemaerweiterung besteht auch die Möglichkeit CAM- bzw. CIM-orientierte Sichten bereitzustellen.

Die hier entwickelte objektunterstützende DB-Schnittstelle läßt sich einerseits nach den zu verarbeitenden Objekten - oben Verarbeitungseinheiten genannt - und andererseits nach den darauf wirkenden Operationen charakterisieren. Die bereitgestellten Objekte als „Gegenstand der Verarbeitung" sind z.T. von weitaus komplexerer Struktur als herkömmliche, flache DB-Tupel. Die operationale Mächtigkeit wird durch drei unterschiedlich mächtige Operationsklassen erreicht. Im Gegensatz zu den satzorientierten Operationen manipulieren die strukturorientierten Operationen immer eine strukturierte Menge heterogener Sätze. Als objektorientierte Operationen bezeichnet man schließlich die Verarbeitung einer ebenfalls strukturierten Menge von heterogenen Sätzen durch eine „semantisch höhere" Aktivität.

Der erreichte Grad an „Objektorientierung" läßt sich vorteilhaft zur Ableitung und Darstellung der Anwendungsobjekte ausnutzen. Ob allerdings noch weitere Maßnahmen zur Objektunterstützung in das DBS integriert werden sollen oder ob das DBS gar die Anwendungsobjekte selbst an seiner „erweiterten" Schnittstelle anbieten soll, muß durch weitere praktische Untersuchungen geklärt werden.

Betrachtet man dagegen den Implementierungs-Overhead, der zur Realisierung dieser objektunterstützenden Datenhaltungsschnittstelle notwendig ist, so stellt man eine deutliche Unangemessenheit des benutzten DBS und der Implementierungssprachen fest. Diese Gründe führten dazu, daß u.a. ein explizites Cursor-Konzept zur objektbezogenen Verarbeitung realisiert werden mußte. Ebenso wurde gezielt Redundanz in das DB-Schema eingebaut, um eine direkte Abbildung der Verarbeitungsobjekte auf die DB-Strukturen zu ermöglichen sowie um zur Verbesserung des Zugriffsverhaltens beizutragen. Andererseits wurde das DB-Schema zur Modellierung von wichtigen Zwischenergebnissen erweitert, die während der Durchführung von Modellierungsoperationen entstehen und mehrfach benötigt werden.

Insgesamt erscheint der hier vorgeschlagene Ansatz einer zugeschnittenen „Objektorientierung" als



```
Polyzyl(n):

ADDBREP ...          Aufbauen eines BREP-Satzes
ADDFACE ...          Aufbauen zweier Face-Sätze, je einer für Deck-
ADDFACE ...          und Bodenfläche des Polyzyls
Loop ADDFACE         Aufbauen eines Face-Satzes für eine Polyzyl-
                     Seitenfläche
    Loop ADDEDGE     Bedingter Aufbau eines Track-, zweier Point- und
                     eines Edge-Satzes sowie Sätze mit den
                     entsprechenden Beziehungen
    End Loop         Abbruch wenn alle Kanten einer Fläche
                     eingebracht sind
    Loop CONFACE     Aufbau der Umgebungsinformation für die
                     gemeinsamen Kanten zweier Flächen
    End Loop         Abbruch wenn die neue Seitenfläche mit der
                     Deck- und Bodenfläche "verbunden" ist
    Loop SET_NBHD_OF_POINT Aufbau der Umgebungsinformation eines
                     Eckpunktes
    End Loop         Abbruch wenn alle "neuen" Punktumgebungen
                     eingebracht sind
End Loop             Abbruch wenn alle Polyzylflächen aufgebaut sind
UPBREP               Aktualisieren der BREP_Hüllenattribute
```

Abb. 9. POLYZYL-Generierungsprozedur und Beispielkörper

Tabelle 5. Aufrufhäufigkeiten beim Aufbau einer POLYZYL-BREP-Darstellung

POLYZYL (n)	5	25	50
Anwendermodell	1	1	1
OSS	64	304	604
Datenmodell (a)	1958	15078	41828
(b)	ca. 1850	ca. 10800	ca. 25600
(c)	ca. 1450	ca. 7400	ca. 14800

Tabelle 6. Aufrufhäufigkeiten beim Aufbau einer POLYZYL-BREP-Darstellung bei redundanzfreiem Schema

POLYZYL (n)	5	25	50
Anwendermodell	1	1	1
OSS	39	179	354
Datenmodell	1179	6159	11484

sehr vielversprechend. Eine bessere Unterstützung durch die Implementierungswerkzeuge (DBS und Programmiersprache) ist allerdings notwendig. Durch die Leistungsanalyse im nachfolgenden Abschnitt wird diese Forderung nachdrücklich untermauert. Zusätzlich wird dort auf weitere nächste Optimierungsvorschläge sowie auf allgemeinere Konsequenzen dieser Analyse eingegangen.

4.3 Analyse einer Beispiel-Anwendermodelloperation

Die Analyse von Aufrufhäufigkeiten an den verschiedenen Ebenen des KUNICAD-Systems dient zum einen einer Aufwandsgrobabschätzung von komplexen Anwendermodelloperationen, zum anderen lassen sich Aussagen über die Tauglichkeit des eingeschlagenen Weges zur Abbildung eines CAD-Anwendermodells auf ein Datenmodell ableiten. Darüberhinaus können Hinweise für andere, bessere Lösungswege gefunden werden. Eine umfassende Analyse wird in [17] durchgeführt.

Hier beschränken wir die Aufwandsbetrachtungen auf die Modelloperation „Generiere eine Ausprägung des Basiskörpers POLYZYL". Durch Veränderung der Anzahl der konstituierenden Seitenflächen ist eine hinreichende Lastvariation möglich. Der Aufwand zur Generierung der strukturellen Information einer Basiskörperausprägung ist äußerst gering, so daß er gegenüber den überraschend hohen Kosten zur Erzeugung der geometrischen Information vernachlässigt werden kann. Der Aufbau der BREP-Struktur erfolgt durch die Operationen der objektunterstützenden Schnittstelle (OSS, vgl. Kapitel 3.4). Abb. 9 enthält neben der graphischen Darstellung eines POLYZYLs den prinzipiellen Aufbau der Generierungsprozedur POLYZYL(n).

Es existieren drei Systemebenen, an denen die Aufrufhäufigkeit interessiert: Anwendermodell-, objektunterstützende und Datenmodell-Schnittstelle. Tabelle 5 gibt Auskunft über konkrete Werte. Es werden die Aufrufhäufigkeiten bei der Generierung eines POLYZYL mit 5, 25 und 50 Seitenflächen gegenübergestellt. Der Aufwand an der OSS-Ebene nimmt proportional mit dem Lastfaktor (Anzahl der Flächen) zu, was als Indiz für einen „natürlichen" Anwendungsbezug dieser Schnittstelle gewertet werden kann. Die Anzahl der Aufrufe an der Datenmodell-Schnittstelle wächst dagegen stark überproportional. Durch eine genauere Untersuchung entsprechender OSS-Operationen lassen sich die folgenden Optimierungsmöglichkeiten aufdecken [17] (vgl. Tabelle 5):

a) Die OSS-Operationen werden *direkt* in Operationen der DBS-Schnittstelle umgesetzt. Durch häufi-

ges Rereferenzieren innerhalb der Verarbeitungseinheit, die von einer OSS-Operation betroffen ist, entsteht eine hohe Zugriffslokalität. Diese kann allerdings nicht explizit, sondern nur unzureichend durch Lokalität im DBS-internen Systempuffer ausgenutzt werden.

b) Innerhalb bestimmter OSS-Operationen wird eine adäquate Hauptspeicherstruktur aufgebaut. Hierdurch wird eine *indirekte* Abbildung einer OSS- in eine Folge von DBS-Operationen möglich. Der Aufbau von Hauptspeicherstrukturen entspricht einer Pufferung von Verarbeitungseinheiten und unterstützt die Nutzung der Zugriffslokalität sehr „nahe" an ihrem „Entstehungsort".

c) Unter Voraussetzung einer gewissen Kenntnis des Verarbeitungszusammenhangs kann der Aufwand in verschiedenen OSS-Operationen weiter reduziert werden. Die Verwaltung dieses Kontextwissens hat in der Modellabbildungsschicht zu erfolgen (also in den Programmen zur Abbildung des Anwendermodells auf die objektunterstützende Schnittstelle). Der erforderliche (eventuell erhebliche) Verwaltungsaufwand geht aus der Aufrufstatistik der Tabelle 5 *nicht* hervor.

Die aufgeführten Werte beruhen auf dem in Kapitel 3.3 vorgestellten DB-Schema. Die Auswirkungen einer möglichen Schemareduktion (redundanzfreies Schema: der Geometrieteil besteht nur aus BREP-, Face-, Edge- und Point-Satztyp) auf die Aufrufhäufigkeiten an den unterschiedlichen Systemschnittstellen ist in Tabelle 6 dargestellt. Die Aufrufhäufigkeiten an der objektunterstützenden sowie an der Datenmodell-Schnittstelle liegen unterhalb der entsprechenden Werte bei Verwendung des redundanzbehafteten Schemas (vgl. Tabelle 5). Da sich die Häufigkeitsangaben auf eine Generierungsfunktion (POLYZYL) beziehen, bei der es im wesentlichen um das Einspeichern von Datensätzen geht, ist dies unmittelbar einleuchtend. Für Modellierungsfunktionen, bei denen der Retrieval- und Update-Anteil dominiert, ist ein gegenteiliger Effekt zu erwarten. Jedoch erscheint auch im vorliegenden Fall bemerkenswert, daß im Vergleich mit den Werten aus dem oben skizzierten, zweiten Optimierungsschritt (Fall c) keine entscheidende Verbesserung erreicht werden kann. Eine wesentliche Verschlechterung muß dagegen für den bislang nicht berücksichtigten Berechnungsaufwand (insbesondere bei der eigentlichen Körpermodellierung) erwartet werden. Obwohl noch keine quantitativen Analysen über den anwendungsbezogenen Mehr- bzw. Minderaufwand vorliegen, erscheint uns diese Einschätzung gerechtfertigt, da die Einführung der kontrollierten Redundanz im DB-Schema ja gerade diesen modellierungsbedingten Rechenaufwand reduzieren sollte. Trotz der skizzierten Optimierungsschritte bleibt die Anzahl der Aufrufe an der Datenmodellschnittstelle enorm hoch. Einer der wichtigsten Gründe hierfür scheint in dem herkömmlichen Subschema-Konzept der DBS-Schnittstelle zu liegen, das den Gegenstand der Verarbeitung auf höchstens eine Satzausprägung pro Satztyp beschränkt. Demgegenüber wird an der OSS in der Regel auf Satzmengen heterogenen Typs operiert.

Soll die Kenntnis des Verarbeitungszusammenhangs besser zum Tragen kommen, so ist ein *Paradigmenwechsel* bzgl. des Darstellungs- und Verarbeitungskonzepts des DBS unumgänglich. Eine weitere Forderung, die hiermit eng zusammenhängt, liegt in der Schaffung *adäquater Datenmodellierungsmöglichkeiten.* Hierbei steht die Definition komplex strukturierter Satzmengen und ihre Manipulation durch entsprechende Datenmodelloperationen im Vordergrund. Eine Diskussion der dazu in der Literatur vorgeschlagenen Modelle findet sich in [6, 15]. Eine einfache Klassifikation dieser Datenmodellvorschläge läßt sich gemäß den zugrundegelegten Modellierungskonzepten finden. Man unterscheidet dabei Konzepte basierend auf abstrakten Datentypen [22, 31] von solchen, die eine „objektorientierte Erweiterung" [3, 4, 19, 26, 27] auf der Basis klassischer Datenmodelle darstellen.

5. Schlußfolgerungen

In dieser Arbeit haben wir ein DB-gestütztes, volumenorientiertes Modellierungssystem für polyederförmige Werkstücke mit folgenden wichtigen Systemeigenschaften vorgestellt:

- Anwendermodell mit abstrakten CAD-Objekten zur adäquaten Unterstützung des Entwurfsprozesses
- Volumenorientiertes Verfahren als Kernalgorithmus zur geometrischen Modellierung
- Zusätzliche interne, automatisch aktualisierte Strukturen nach dem Begrenzungsflächenmodell zur effizienten graphischen Repräsentation der Werkstücke.
- Bereitstellung einer objektunterstützenden DB-Schnittstelle zur Vereinfachung der Abbildung der Anwenderobjekte.

Ist die Realisierung des KUNICAD-Systems aus der Sicht der Funktionalität und des Software Engineering durchaus zufriedenstellend, so kann sie erwartungsgemäß in ihrem Leistungsverhalten nicht überzeugen. Es lag jedoch überraschender Weise um Größenordnungen über dem Aufwand, der in einer interaktiven Umgebung in Kauf genommen werden kann. Erste Maßnahmen wie

- Verbesserung der Lokalität der Verarbeitung von Datenstrukturen in der Anwendungskomponente
- Ausnutzung von Anwendungswissen bei der Abbildung der Modellierungsalgorithmen

konnten den Aufwand um den Faktor 3 reduzieren. Trotz Nutzung dieses Optimierungspotentials ist noch ein erheblicher Leistungssprung zu erzielen, bevor DB-gestützte CAD-Systeme in der Praxis eingesetzt werden können. Die Vereinfachung des Schemas - durch Entfernung gewisser Redundanzen - erscheint uns kein geeigneter Lösungsweg zu sein, da dann viele Sachverhalte nur implizit dargestellt und die Modellierungsalgorithmen komplizierter werden. Eine zentrale Rolle bei solchen Optimierungsüberlegungen muß dagegen das Einbettungskonzept der DB-Daten in die Anwendung spielen. Da Anwendungsobjekte sich aus einer Vielzahl von Satztypen mit jeweils großer Anzahl von Ausprägungen zusammensetzen, muß eine Möglichkeit gefunden werden, heterogene Satzmengen in einer flexiblen und der Anwendung verständlichen Weise in die Anwendung einzubinden. Ob die geeignete Form der Einbingung „objektgekapselt" ist, d.h., der Anwendung stehen ausschließlich vordefinierte Operatoren (ADT's) zur Verfügung, oder ob sie „strukturorientiert" mit direkter Manipulationsmöglichkeit der Anwendung ist - oder gar eine Mischform aus beiden -, muß in weiteren Forschungsarbeiten geklärt werden. Dabei ist auch die Frage zu untersuchen, ob jede Aktualisierungsoperation der Anwendung direkt dem DBS mitgeteilt wird oder ob es möglich ist, Folgen von Operationen lokal durchzuführen und ihre „Ergebnisse" zusammengefaßt dem DBS zu übergeben.

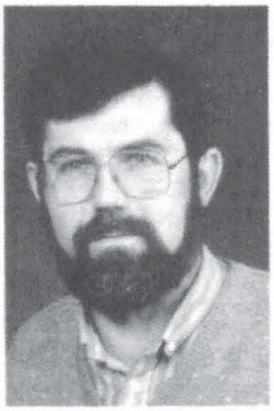
Die bei der Analyse des KUNICAD-Systems gemachten Beobachtungen konnten wir durch ähnliche Untersuchungen von DB-gestützten Anwendungen aus dem Bereich des VLSI-Entwurfs [33] und der geographischen Datenverarbeitung [28] untermauern. Alle Ergebnisse deuten darauf hin, daß geeignete „Objektschnittstellen" für das DBS benötigt werden, wobei der dabei anzubietende Grad an Objektunterstützung noch offen und möglicherweise durch die Art der Anwendung vorbestimmt ist. Ganz sicher müssen jedoch solche „Objektbanksysteme" oder DBS für Non-Standard-Anwendungen [16] wesentlich ausdrucksmächtigere Datenmodelle als herkömmliche DBS besitzen.

Danksagung. Diese Arbeit entstand im Rahmen des Projektes D2 des von der Deutschen Forschungsgemeinschaft geförderten Sonderforschungsbereichs 124. Zahlreiche Diskussionen mit W.E. Fischer (Philips Forschungslabor Hamburg) und A. Meier (ETH Zürich) halfen uns wesentlich bei der Konzeptfindung und Datenmodellierung. Wir danken den Referenten, die uns nach gründlicher Lektüre der ersten Fassung wertvolle Hinweise zur Verbesserung gegeben haben.

Literatur

1. ACM-Siggraph Commitee: Final report of the GSPC state of the art subcommittee. Computer Graphics **12**, 1/2 (1978)
2. Barsky, B.A.: Computer-aided geometric design. A bibliography with keywords and classified index. Computer Graphics **16**, 67-159 (1982)
3. Batory, D.S., Buchmann, A.P.: Molecular objects, abstract data types and data models: A framework. In: Proc. of 10th Int. Conf. on Very Large Data Bases, Singapore, pp. 172-184. Saratoga, Calif.: VLDB-Endowment 1984
4. Dadam, P., et al.: A DBMS prototype to support extended NF^2 relations: An integrated view on flat tables and hierarchies. In: Proc. of Int. ACM SIGMOD Conf., Washington D.C. 1986
5. Deisel, W.: Mengenoperationen auf Polyedern. Studienarbeit, IMMD VI. Univ. Erlangen-Nürnberg 1982
6. Dittrich, K.R., Kotz, A.M., Mülle, J.A., Lockemann, P.C.: Datenbankunterstützung für den ingenieurwissenschaftlichen Entwurf - eine Übersicht über den Stand der Forschung. Informatik Spektrum **8**, 112-125 (1985)
7. Eberlein, W.: CAD-Datenbanksysteme - Architektur technischer Datenbanken für integrierte Ingenieursysteme. Berlin, Heidelberg, New York, Tokyo: Springer 1984
8. Encarnacao, J. et al.: A reference model for components and interfaces of CAD-systems. In: Proc. of IFIP WG 5.2/WG 5.3 Working Conference on Integration of CAD/CAM, Dresden, pp 135-153. Amsterdam, New York, Oxford: North-Holland 1984
9. Fischer, W.E.: Datenbanksystem für CAD-Arbeitsplätze, Informatik-Fachberichte 70. Berlin, Heidelberg, New York, Tokyo: Springer 1983
10. Proc. GI-Fachtagung Geometrisches Modellieren. Informatik-Fachberichte 65. Berlin, Heidelberg, New York, Tokyo: Springer 1983
11. International Standard Organisation (ISO): Graphical Kernel System (GKS) Version 7.2. New York: ISO 1982 (Dok.-Nr. ISO/DIS 7942)
12. Grätz, J.-F.: Modellalgorithmen zur dreidimensionalen Geometriefestlegung komplexer Bauteile mit beliebiger Flächenbegrenzung in der rechnerunterstützten Konstruktion. Diss., Bochum 1983
13. Härder, T., Hübel, Ch., Mitschang, B.: Use of inherent parallelism in database operations. In: Proc. of Conference on Algorithms and Hardware for Parallel Processing CONPAR 86, Aachen, Lecture Notes in Computer Sciences. Berlin, Heidelberg, New York, Tokyo: Springer 1986
14. Hübel, Ch., Langenfeld, S.: Datenbankbasierter 3D-Bauteilmodellierer für Werkstücke. Diplomarbeit, Universität Kaiserslautern 1985
15. Hartzband, D.J., Maryanski, F.J.: Enhancing knowledge representation in engineering databases. IEEE Computer **18**, 39-48 (1985)
16. Härder, T., Reuter, A.: Architektur von Datenbanksystemen für Non-Standard-Anwendungen. In: Proc. GI-Fachtagung Datenbank-Systeme in Büro, Technik und Wissenschaft, Informatik-Fachberichte 94, S. 253-286 (eingeladener Vortrag). Berlin, Heidelberg, New York, Tokyo: Springer 1985
17. Hübel, Ch.: Aufwandsanalyse der Datenbankfunktionen bei einem geometrischen Modellierungssystem, Forschungsbericht, SFB 124, Universität Kaiserslautern 1986
18. Initial graphics exchange specification (IGES), Version 2.0, U.S. Department of Commerce, NBS, Washington, DC 1983
19. Katz, R.H.: Information management for engineering design. Surveys in computer science. Berlin, Heidelberg, New York, Tokyo: Springer 1985
20. Kim, W., Lorie, R.A., McNabb, D., Plouffe, W.: A transaction mechanism for engineering design databases. In: Proc. 10th

Int. Conf. on Very Large Data Bases, Singapore, pp 355–362. Saratoga, Calif.: VLDB-Endowment 1984
21. Lockemann, P.C. et al.: Anforderungen technischer Anwendungen an Datenbanksysteme. In: Proc. GI-Fachtagung Datenbank-Systeme für Büro, Technik und Wissenschaft, Informatik-Fachberichte 94, S 1–26. Berlin, Heidelberg, New York, Tokyo: Springer 1985
22. Lüke, B.: DANTE – Ein semantisches Datenmodell für Anwendungen aus dem Konstruktionsbereich, Interner Bericht, Universität Karlsruhe, 1983
23. Mayer, A.: Graphischer Editor für reguläre Polyedergebilde. Diplomarbeit, ETH Zürich 1984
24. Meier, A.: Applying relational database techniques to solid modeling. In: Proc. GI-Fachtagung Datenbanksysteme in Büro, Technik und Wissenschaft, Informatik-Fachberichte 94, Seite 50–67. Berlin, Heidelberg, New York, Tokyo: Springer 1985
25. Mitschang, B.: Überlegungen zur Architektur von DB-Systemen für Ingenieuranwendungen. In: Tagungsband der 14. GI-Jahrestagung, Informatik-Fachberichte 88, S 318–334. Berlin, Heidelberg, New York, Tokyo: Springer 1984
26. Mitschang, B.: Ein Datenmodell zur Abbildung von komplexen Objekten. Forschungsbericht des SFB 124, Nr. 20/85, Universität Kaiserslautern 1985
27. Schek, H.-J.: Towards a basic relational NF^2 algebra processor. In: Proc. 2nd Int. Conf. on Foundations of Data Organization, Kyoto 1985
28. Reuter, J., Schulz, M.: DGBEO – datenbankbasiertes Kernsystem eines Geographischen Informationssystems. Diplomarbeit, Universität Kaiserslautern 1986
29. Requicha, A.A.G., Voelcker, H.B: Solid modelling: A historical summary and contemporary assessment. IEEE Computer Graphics Appl. **2**, 9–24 (1982)
30. Requicha, A.A.G., Voelcker, H.B.: Boolean operations in solid modelling: Boundary evaluation and merging algorithms. Technical Memorandum No. 26, Production Automation Project, University of Rochester, New York, 1984
31. Stonebraker, M., Guttman, A.: Using a relational database management system for computer aided design data – an update. IEEE Database Eng. **7**, 56–60 (1984)
32. Technische Unterlagen zum Graphiksystem SIGMEX 6161, Sigmex GmbH München, 1986
33. Sottong, W.: Datenstrukturen im VLSI-Schaltungsentwurf. Diplomarbeit, Universität Kaiserslautern 1985
34. UDS, verschiedene Handbücher zum Datenbanksystem der Firma Siemens, München, 1984



Theo Härder (40), Studium der Elektrotechnik an der TH Darmstadt (1966–1971), seit 1972 Mitarbeit im Fachbereich Informatik (Datenverwaltungssysteme) der TH Darmstadt, 1975 Promotion, 1976 Post-doctoral Fellow in IBM Research, San Jose (Mitarbeit im Projekt „System R"), 1977 Dozent, 1978 Professor für Informatik an der TH Darmstadt, seit 1980 Professor für Praktische Informatik (Datenverwaltungssysteme) an der Universität Kaiserslautern. Besondere Forschungsinteressen: Leistungsanalyse von DB- und DB/DC-Systemen, Mehrrechner-Datenbanksysteme, Non-Standard-Anwendungen für DBS (CAD/CAM), Datenbanken in Netzen.

Christoph Hübel (27), Studium der Informatik an der Universität Kaiserslautern (1979–1985) mit Schwerpunkt Praktische Informatik (Datenverwaltungssysteme) und Nebenfach Elektrotechnik. Wissenschaftlicher Mitarbeiter im Sonderforschungsbereich 124 am Fachbereich Informatik der Universität Kaiserslautern. Hauptarbeitsgebiet: DBS-Unterstützung für Non-Standard-Anwendungen (CAD/CAM), anwendungsorientierte DBS-Schnittstellen.

Stefan Langenfeld (26), Studium der Informatik an der Universität Kaiserslautern (1978–1985) mit Schwerpunkt Praktische Informatik (Software-Technik) und Nebenfach Wirtschaftswissenschaften. Zur Zeit Mitarbeiter der Lufthansa AG, Flughafen Frankfurt/Basis.

Bernhard Mitschang (27), Studium der Informatik an der Universität Kaiserslautern (1977–1982) mit Schwerpunkt Praktische Informatik (Datenverwaltungssysteme) und Nebenfach Mathematik. Wissenschaftlicher Mitarbeiter im Sonderforschungsbereich 124 am Fachbereich Informatik der Universität Kaiserslautern. Hauptarbeitsgebiet: Datenmodelle für Non-Standard-Anwendungen, Architektur von Non-Standard-Datenbanksystemen.