Datenbankunterstützung für ein CAD-Arbeitsplatzsystem

H.-P. Christmann, K. Meyer-Wegener, B. Mitschang, A. Sikeler
Universität Kaiserslautern
Fachbereich Informatik
Postfach 30 49
6750 Kaiserslautern

Uberblick

Die stetige Erweiterung und Integration von Anwendungen aus dem Bereich der graphischen Datenverarbeitung verursacht ein schnelles Wachstum von Umfang und Komplexität der zu verwaltenden Datenbestände und erfordert daher in steigendem Maße Datenbankunterstützung. Für die adäquate Unterstützung in einem CAD-Arbeitsplatzsystem ist die zugrunde gelegte Rechnerkonfiguration des Arbeitsplatzes, das zu verwendende Datenbanksystem und die Zusammenarbeit des Datenbanksystems mit dem eigentlichen CAD-System von großer Wichtigkeit. Hier werden nun zum einen Anforderungen an geeignete Datenbanksysteme vorgestellt und zum anderen, nach einer kurzen Diskussion über passende Rechnerkonfigurationen, einige Vorschläge für die Zusammenarbeit von Graphik- und Datenbanksoftware angegeben.

1. Einleitung

Die ersten Programmsysteme aus dem Bereich der graphischen Datenverarbeitung (z.B. CAD/CAM) setzten sowohl bezüglich der graphischen Ein-/Ausgabe als auch bezüglich der Datenhaltung noch auf sehr niedrigen Schnittstellen auf: manchmal direkt auf der Geräteschnittstelle bzw. auf der üblichen Betriebssystem-Dateischnittstelle. Sie waren dadurch extrem auf die zugrundeliegende Hard- und Software zugeschnitten, was sie einerseits sehr effizient, andererseits aber auch äußerst änderungsempfindlich und nicht portabel machte. Jedes Programmsystem stellte eine Insellösung für eine spezielle Teilaufgabe dar, z.B. für Zeichnungserstellung, Produktentwicklung, 3D-Modellierung, NC-Programmierung etc. Das Gesamtsystem bestand also aus einzelnen voneinander

abgegrenzten Teilsystemen, die jeweils ihre eigene Datenhaltung mit Hilfe von separaten Dateien abwickelten.

Die stetige Erweiterung der graphischen Datenverarbeitung verursachte u.a. ein schnelles Wachstum der zu verwaltenden Datenbestände und eine steigende Komplexität der graphischen Ausgabe auf die angeschlossenen heterogenen (graphischen) E/A-Geräte.

Mit der Entwicklung und Standardisierung des Graphischen Kernsystems (GKS, siehe /GKS83/) wurde zunächst Geräteunabhängigkeit und damit Portabilität bezüglich graphischer Systeme erreicht.

Parallel zur Entwicklung von GKS wurde versucht, auch in bezug auf die Datenhaltung eine größere Unabhängigkeit zu erlangen. Der erste Schritt dazu war die Einführung einheitlicher Datenstrukturen und Dateiformate (z.B. IGES, siehe /En84/), die zwar den Datenaustausch zwischen Anwendungsprogrammen (AP) unterstützten, aber weiterhin keinerlei Konsistenzsicherungsmaßnahmen sowie nur eine geringe Unabhängigkeit von der Datenhaltung und dem AP gewährleisteten. Dies und auch die ständig fortschreitende Expansion und Integration der AP führten zum Einsatz konventioneller Datenbanksysteme (DBS). Damit wurden einerseits die Probleme der Änderungsempfindlichkeit und Konsistenzerhaltung zwar verringert, andererseits taten sich aber neue schwerwiegende Probleme auf: Umständlichkeit in der Datenmodellierung und Anfragestellung sowie deutliche Performance-Verluste gegenüber den früheren Spezialsystemen (siehe /Mi84/).

Es gibt zur Zeit eine ganze Reihe von neuen Lösungsansätzen, die von der Entwicklung spezieller DBS für graphische Anwendungen ausgehen. Im folgenden Abschnitt werden zunächst die Ideen, die hinter solchen "Non-Standard-DBS" (NDBS) stehen, etwas allgemeiner erläutert. Danach wird eine Rechnerkonfiguration für ein solches NDBS vorgestellt, bei der an einen graphischen Arbeitsplatzrechner ein spezieller DB-Rechner angeschlossen ist, der ggf. mit dem DBS in einem großen Zentralrechner kommuniziert. Im dritten Abschnitt werden schließlich zwei Vorschläge für die Zusammenarbeit von Graphik- und Datenbanksoftware auf diesen Rechnern diskutiert.

2. Einsatz von Non-Standard-Datenbanksystemen im CAD-Bereich

Die o.g. Problempunkte führten vielerorts zu Überlegungen, neue, auf bestimmte Anwendungen zugeschnittene Non-Standard-DBS (siehe /HR83/, /Mi84/) zu entwickeln. Dabei besteht das Ziel darin, die anwendungsspezifische Sicht des Benutzers/AP so modellieren zu können, daß möglichst kein Informations- und Bedeutungsverlust auftritt (semantische Verlustfreiheit). Zum anderen muß das zugrundeliegende NDBS auch eine optimale und damit effiziente Verwaltung des modellierten Schemas durchführen.

2.1 Anwendungsbeispiel

Am Beispiel eines datenbankgestützten geometrischen Modellierungssystems (GMS) für dreidimensionale Objekte (mit einfacher Graphikschnittstelle) soll im folgenden der konzeptionelle Unterschied zwischen einem NDBS und einem herkömmlichen DBS aufgezeigt werden.

Die meisten GMS finden ihre Anwendung in der rechnergestützten Konstruktion und Fertigung von Bauteilen und bilden dort das Kernstück von CAD-Systemen. Die angebotenen Funktionen dienen zum Aufbau, zur Speicherung und zur Veränderung von 3D-Objekten. Ein GMS kann in drei Teilkomplexe unterteilt werden:

- benutzergerechtes Anwendermodell zur Mensch-System-Kommunikation
- Schema für die rechnerinterne Darstellung der zu verwaltenden 3D-Objekte
- Geometrieprozessor bestehend aus den notwendigen Algorithmen zum Ausführen der Anwendermodelleingabe auf der zugrundeliegenden Datenstruktur.

Im praktischen Einsatz haben sich das CSG-Modell (Volumenmodell, siehe /Re80/) als Anwendermodell und das Begrenzungsflächenschema (BREP, siehe /Re80/) als rechnerinterne Darstellung bewährt. Deshalb wird dieses hybride Darstellungsschema auch für unser Beispiel gewählt. Dem (Standard-)Volumenmodell liegt die Vorstellung zugrunde, komplexe Körper aus einfachen Standardvolumina, sog. Primitiven (Primitivkörpern), mit Hilfe von Mengenoperationen zu kombinieren. Im Gegensatz dazu stellt die

Begrenzungsflächendarstellung einen Körper durch die explizite Angabe der ihn begrenzenden Flächen, Kanten und (Eck-)Punkte dar.

Die an der Anwenderschnittstelle damit angebotenen Funktionen sind u.a.:

- körpergenerierende Operatoren, wie ERZEUGEN eines Primitivkörpers
- körpermanipulierende Operatoren, wie ZUSAMMENSETZEN, SCHNEIDEN, SUBTRAHIEREN zweier Körper sowie TRANSLATION und ROTATION eines Körpers
- verwaltende Operatoren, wie KOPIEREN, LÖSCHEN, HOLEN, ABSPEICHERN, INFORMIEREN
- graphische Operatoren, wie PLOTTEN und AUSGEBEN.

Zerlegt man nun die Algorithmen des Geometrieprozessors, unter Beachtung der im Anwendermodell angebotenen Operatoren, in "einfachere" Operationen auf der rechnerinternen Darstellung, so kristallisieren sich u.a. die folgenden häufig auszuführenden Operationen heraus:

- Zerlegung eines Körpers in die zugehörigen Flächen (bzw. Kanten),
- Schnitt von Fläche (Kante) mit Fläche,
- Verbinden von Punkten zu Kanten, bzw. von Kanten zu Flächen und von Flächen zu Körpern,
- Test auf Enthaltensein, Parallelität bzw. Kollinearität und Überschneidung sowie
- Abstands- und Winkelberechnungen.

Diese Teiloperationen lassen deutlich den engen Bezug zur rechnerinternen Darstellung erkennen. Daraus folgt für ein NDBS, daß zusammen mit der benutzergerechten Datenmodellierung auch die auf den Daten auszuführenden Operationen bereitgestellt werden sollten. Hierdurch wird also die Forderung nach einer Art "anwendungsorientiertem abstrakten Datentyp" aufgestellt.

Eine der im Relationenmodell möglichen Modellierungen sieht nun wie folgt aus:

- Relation KÖRPER (KÖRPER_NR,...)
- Relation FLÄCHE (FLÄCHEN_NR, ..., KÖRPER_NR)
- Relation FLÄ_KAN (FLÄCHEN_NR, KANTEN_NR)
- Relation KANTE (KANTEN_NR, ...)
- Relation KAN_PUN (KANTEN_NR, PUNKT_NR)

- Relation PUNKT (PUNKT_NR, X_KOORD, Y_KOORD, Z_KOORD, ...).

Dies ist eine nicht redundante Modellierung der Hierarchie, die bei der Darstellung eines 3D-Objekts im Begrenzungsflächenmodell entsteht. Sie enthält alle für die Algorithmen des Geometrieprozessors notwendigen Daten. Versucht man allerdings die oben aufgelisteten Operationen mit den vom Relationenmodell zur Verfügung gestellten Anfrageoperatoren auf diesem Relationenschema auszudrücken, so stellt man sehr schnell die damit verbundene Umständlichkeit und Komplexität fest. Einige ausgewählte Teiloperationen (s.o.) sehen unter Benutzung der Anfragesprache SQL (siehe /Da81/) dann wie folgt aus:

- Zerlegung eines Körpers in seine zugehörigen Flächen und Kanten

```
SELECT  E.KANTEN_NR, F.FLÄCHEN_NR, K.KÖRPER_NR
FROM  KÖRPER K, FLÄCHE F, KANTE E, FLÄ_KAN FE
WHERE  K.KÖRPER_NR = '999'   AND                (* Selektion *)
       K.KÖRPER_NR = F.KÖRPER_NR   AND           (* Join *)
       F.FLÄCHEN_NR = FE.FLÄCHEN_NR   AND        (* Join *)
       FE.KANTEN_NR = E.KANTEN_NR                (* Join *)
```

- Test, ob sich zwei Flächen überschneiden bzw. berühren
  - Test, ob beide Flächen in der gleichen Ebene liegen (mittels Parallelitätstest)
  - falls ja: .. Holen der Kantenauflösung jeder Fläche (1 Selektion und 2 Join)
    .. Verschneiden jeder Kante der einen Fläche mit jeder der anderen Fläche
  - falls Schnittmenge nicht leer, dann Überschneidung oder Berührung

Schaut man sich diese Modellierungs- und Anfrageergebnisse an, so erkennt man, daß Relationen definiert wurden, die anstatt ein 3D-Objekt zu beschreiben, jeweils nur Teilaspekte desselben darstellen. Das ursprüngliche Entity "3D-Objekt" ist verschwunden. Stattdessen mußten Relationen (FLÄ_KAN und KAN_PUN) hinzugefügt werden, die in dem Darstellungsschema gar nicht vorhanden sind. Dies bedeutet, daß die vom zugrundeliegenden Darstellungsschema (BREP) realisierte Abstraktionsebene a priori nicht existiert. Die gewünschten Darstellungen müssen jeweils aus den vorhandenen Teilstrukturen rekombiniert werden, was extrem aufwendig erscheint. Zusätzlich müssen die Operationen der gültigen Modellierung und den verfügbaren Modelloperatoren angepaßt

werden. Analog zu dieser Operationsmodifikation muß auch eine Anpassung der auszuführenden Integritätskontrollen erfolgen.

Das hier aufgeführte Beispiel zeigt deutlich, welche Ziele durch den Einsatz von NDBS erreicht werden sollen und definiert dadurch Forderungen an ein NDBS:

- ein anwendungsbezogenes Datenmodell, das die Natürlichkeit und Einfachheit der Darstellung und die Formulierbarkeit aller Integritätsbedingungen gewährleisten sowie entsprechend höhere Operationen anbieten sollte,
- optimale Realisierung des Datenmodells durch das zugrundeliegende System.

Hierzu sind beispielsweise angepaßte Speicherungsstrukturen und Zugriffspfade notwendig, die neuartige, in den Anwendungen vorkommende Zugriffsoperationen, wie bei räumlichen oder nachbarschaftsbezogenen Anfragen, unterstützen (siehe /Mi84/).

2.2 Architekturvorschläge für NDBS

In großem Maße ausschlaggebend für die Leistungsfähigkeit eines DBS allgemein und auch eines Non-Standard-DBS ist der gewählte Architekturansatz. In /Mi84/ werden einige Architekturvorschläge für mögliche NDBS angegeben und bewertet. Die dort vorgestellten Architekturansätze reichen von einer einfachen Zusatzebenen-Architektur, die dadurch gekennzeichnet ist, daß eine anwendungsbezogene Abbildungsschicht auf ein herkömmliches DBS aufgesetzt wird, bis zu einer Datenbanksystemkern-Architektur, die zweigeteilt ist und aus einem anwendungsunabhängigen Kernsystem und einer anwendungsorientierten Modellabbildungsschicht besteht. Das Kernsystem, Speicherserver genannt, bietet nun die Möglichkeit, bestimmte und für die meisten Anwendungen wichtige Eigenschaften gezielt zu unterstützen. Hingegen erlaubt die Modellabbildung, ein anwendungsorientiertes Modellierungswerkzeug anzubieten, und setzt die so definierte Sicht auf die Speicherserver-Schnittstelle um.

Dieser Architekturansatz scheint die oben aufgestellten Forderungen an ein NDBS relativ direkt und damit einfach zu erfüllen und wird daher den

folgenden Betrachtungen zugrundegelegt.

2.3 Ein Rechner für das NDBS: das DATABASE-FRONT-END

Zur Realisierung der im vorigen Paragraph erwähnten Kernarchitektur für ein NDBS bieten sich mehrere Möglichkeiten an:

- Einbettung des NDBS in einen zentralen Verarbeitungsrechner oder
- Einbettung des NDBS in den Arbeitsplatzrechner (APR) oder
- Bereitstellung des NDBS auf einem eigenen, dem APR zugeordneten Rechner.

Im ersten Fall stellt das NDBS eine von vielen Anwendungen im zentralen Rechner dar und muß sich daher dessen Ressourcen auch mit allen anderen teilen. Nachteilig kann sich auch die Abhängigkeit von den Verbindungen zum und vom zentralen Rechner selbst auswirken. Bei Störungen oder anderen Ausfallzeiten ist in diesem Falle ein lokales Arbeiten mit dem Arbeitsplatzsystem nicht mehr möglich, da der Anschluß an die Datenbank fehlt. In den beiden anderen Ansätzen wird daher das NDBS dem Arbeitsplatzsystem direkt zugeordnet. Damit entfällt weitgehend die Abhängigkeit vom Zentralrechner, und ein lokales Arbeiten wird ermöglicht. Bei einigen Arbeitsplatzsystemen, die auf bestimmte Anwendungsgebiete ausgerichtet sind, ist die frei verbleibende Rechen- und Speicherkapazität jedoch zu gering, um das NDBS noch in das System einzubetten. In diesem Fall empfiehlt es sich, für das NDBS einen eigenen Rechner einzusetzen, der dann auch gänzlich auf die Erfordernisse des NDBS zugeschnitten werden kann (/Ch84/). Weitere Vorteile sind die schon erwähnte Abkopplung des Arbeitsplatzes vom zentralen Rechner und die potentiell hohe Leistungsfähigkeit der Kombination eines NDBS mit einem darauf zugeschnittenen Rechner - dem Database-Front-End (DBFE). Die Leistungsfähigkeit dieses Systems beruht unter anderem auf einem an die Anforderungen des NDBS angepaßten Betriebssystem, dem speziellen NDBS selbst und letztlich auch auf den Möglichkeiten zur Parallelverarbeitung innerhalb des DBFE.

3. Die Zusammenarbeit von Graphik- und DB-Software auf Arbeitsplatzrechnern

Der Anschluß eines DBS an ein CAD-System dient, wie in der Einleitung bereits angedeutet, im wesentlichen zwei Zielen:

- Vereinheitlichung der Datenhaltung innerhalb des CAD-Systems (Erhöhung der Portabilität) zusammen mit der Möglichkeit, größere Datenmengen zu speichern
- Erweiterung der gespeicherten Daten um nichtgraphische Information (z.B. Materialeigenschaften)

Es sollen sich also sowohl die Quantität als auch die Qualität der gespeicherten Daten erhöhen. Zur Beantwortung der Frage, wie der Anschluß erfolgen soll, wird zunächst die Strukturierung eines CAD-Systems betrachtet. Ein Beispiel für eine solche Strukturierung geben Anderl et al. in /GKS83/:

- Rechnerinternes Modell (plus Datenhaltungskomponente)
  (z.B. 2D-Kantenmodell, 3D-Kantenmodell, Begrenzungsflächenmodell, Volumenmodell)
- Methodenbaustein
  (z.B. Geometrie, Berechnungen, NC-Steuerdatenerzeugung)
- Kommunikationsbaustein
  (z.B. Anwendungsgraphik, Sprachinterpreter, Ablaufsteuerung)
  mit GKS als Basis

Die anwendungsorientierten Bestandteile eines CAD-Systems (in der obigen Strukturierung sind dies der Methoden- und der Kommunikationsbaustein), werden im folgenden unter der Bezeichnung Anwendungsprogramm (AP) zusammengefaßt (Abb. 1). Anhand dieser Strukturierung sollen nun grundsätzliche Möglichkeiten zur Unterstützung eines CAD-Systems durch ein DBS bzw. NDBS diskutiert werden.

3.1 Allgemeine Möglichkeiten der DB-Unterstützung von CAD-Systemen

Eine sehr einfache Möglichkeit der Unterstützung eines CAD-Systems durch

ein DBS ist die lose Kopplung über Umsetzprogramme (/CAD84/), die Daten aus dem CAD-System (das rechnerinterne Modell) lesen und in der DB abspeichern bzw. umgekehrt (Abb. 2). Das DBS kann dabei als reiner Archivspeicher dienen, in dem die graphischen Daten zu einem Objekt als unstrukturierte Einheit ("Konserve") abgelegt werden können. Es kann sich auch auf die nichtgraphischen Zusatzinformationen beschränken. Die Umsetzprogramme sind dann erheblich komplexer und enthalten anwendungsspezifisches Wissen (etwa über die Beziehungen zwischen der räumlichen Gestalt und den mechanischen Herstellungs- und Bearbeitungstechniken). Drittens schließlich - und diese Lösung wird mit den in Abschnitt 2 vorgestellten NDBS angestrebt - kann das DBS beides aufnehmen: räumliche Information, die sich in eine graphische Darstellung umsetzen läßt, und die den technischen Objekten zugeordnete nichträumliche Information. Entsprechend umfangreich müssen dann natürlich auch die Funktionen der Umsetzprogramme sein.

Der Ansatz der losen Kopplung mit Hilfe von Umsetzprogrammen bietet sich vor allem bei CAD-Systemen an, die keinen direkten Zugriff auf die Modelldaten und auch keine Erweiterung des Anwendungsprogramms erlauben (geschlossene Systeme). Bei offenen Systemen hingegen, bei denen das Anwendungsprogramm erweitert werden darf, kann zum AP des CAD-Systems ein direkter Zugriff auf die Datenbank hinzugefügt werden (Abb. 3). Es bleibt allerdings bei der getrennten Datenhaltung im CAD-System und im DBS. Für die Teilaufgabe des DBS sind wieder die drei Lösungen denkbar, die oben bei der Kopplung mit Umsetzprogrammen genannt wurden. Der Weg zwischen CAD-System und DBS ist wesentlich kürzer und unterliegt nicht mehr der Kontrolle des Benutzers (der die Umsetzprogramme explizit aufrufen mußte), sondern allein der des AP. Dort steht auch das erforderliche Anwendungswissen zur Verfügung, das für die Entscheidung benötigt wird, wann und wie oft ein Datenaustausch zwischen dem CAD-System und dem DBS erfolgen muß. Dafür wird nach wie vor ein Vermittler gebraucht (hier das AP), so daß wir wieder nur von loser Kopplung sprechen wollen (in Abweichung von /CAD84/). Die funktionale Komplexität der Umsetzprogramme (Transformation der Daten) bleibt erhalten und wird Teil des AP.

Bei der dritten Möglichkeit zur Anbindung des DBS an ein CAD-System ist diese Umwandlung nicht mehr erforderlich. Hier wird ein NDBS, wie es in

Abschnitt 2 vorgestellt wurde, direkt in das CAD-System eingebettet (Abb. 4); es übernimmt einheitlich alle Datenhaltungsfunktionen. Dadurch können die Vorteile, die das NDBS gegenüber einem kommerziellen DBS bietet, voll ausgenutzt werden. In den folgenden beiden Abschnitten sollen zwei dieser Einbettungen näher diskutiert werden.

3.2 Integration von graphischem DBS und GKS

Aus der Sicht des Anwendungsprogramms ist es sicher besser, es nur mit einem einzigen (Basis-) System oder Unterprogrammpaket zu tun zu haben, das ein breites Spektrum von Funktionen sowohl zur graphischen Ein- und Ausgabe als auch zur (graphischen) Datenhaltung anbietet. Man kann sich dabei eine gemeinsame Abbildungsschicht oberhalb von GKS und dem (graphischen) DBS vorstellen, die eine Erweiterung der "anwendungsorientierten Schicht" von Anderl et al. (/GKS83/) im Hinblick auf die Verwaltung der Modelldaten darstellt (Abb. 5).

Diese Abbildungsschicht - wir wollen sie Graphisches Darstellungs- und Speicherungssystem (GDSS) nennen - kann (z.B. aus Kompatibilitätsgründen) zunächst alle Funktionen von GKS und alle Funktionen des DBS einfach an den Benutzer durchreichen. Da sie aber die Kontrolle über beide Systeme, GKS und NDBS, hat, kann sie zusätzlich die Konsistenz von graphischer Darstellung auf den Ausgabegeräten und der Darstellung in der DB überwachen. Das äußert sich dann beispielsweise in einigen neuen Returncodes. Nach einer Graphik-Operation kann das GDSS etwa die Warnung "Die Darstellung auf dem Bildschirm entspricht nicht mehr dem Objekt in der DB" zurückliefern. Dieses Objekt sollte daher später ggf. auch in der DB noch modifiziert werden. Das gleiche gilt umgekehrt für einen Änderungsaufruf an das DBS. Ganz verbieten bzw. zurückweisen sollte das GDSS diese Operationen aber wohl nicht, um das AP in seinen Modellierungsmöglichkeiten nicht zu stark einzuschränken.

Das Problem einer möglichen Inkonsistenz tritt nicht mehr auf, wenn das GDSS neue, mächtigere Operationen anbietet, die auf beide Systeme wirken. Ein Beispiel dafür wäre das Aufsuchen eines Objekts in der DB, die Übergabe der nichtgraphischen Information an das AP und die direkte Ausgabe der graphischen Darstellung über GKS - alles mit einem einzigen

GDSS-Befehl. Wünscht der Benutzer eine Translation oder Rotation des Objekts, könnte diese Operation mit anderen Darstellungsparametern wiederholt werden. Das DBS kann die Lokalität der Zugriffe zur Performance-Verbesserung nutzen. Es gibt übrigens auch Vorschläge, diese Art von Operationen direkt an den Benutzer durchzureichen (/Si84/). Er formuliert dann z.B.:

```
DEFINE VIEWPORT name1 WITH ... ;

SELECT P.PARTNO, P.SUPPLIER, GRAPHIC (P) ON name1
FROM   PARTS P
WHERE  P.NAME = 'Kotflügelblech';
```

Mit DEFINE VIEWPORT legt der Benutzer fest, auf welchen Teil seines Bildschirms die graphische Darstellung des Objekts ausgegeben werden soll. Mit der SELECT-Anweisung erfolgt der Zugriff auf die Datenbank, aus der die graphische Darstellung des gewünschten Teils (GRAPHIC) und einige Zusatzangaben wie Teilnr. und Lieferant entnommen werden.

Die umgekehrte Vorgehensweise, also die Erzeugung eines Objekts in der DB aus einer eingegebenen graphischen Darstellung, wird sich dagegen nicht so einfach in einer einzigen Operation zusammenfassen lassen. Dies liegt zum einen an den nichtgraphischen Zusatzinformationen und zum anderen an der Mehrdeutigkeit graphischer Darstellungen (technische Zeichnungen, 2D/3D-Problematik, siehe z.B. /SG83/). Hier bedarf es interaktiver Interpretationshilfe durch den Benutzer.

Ein Konstruktionsschritt, der aus existierenden Objekten neue schafft, wird dagegen wieder sehr gut unterstützt. Dabei muß auf jeden Fall auf die Datenbank zugegriffen werden. Auch in der DB (nicht nur am Bildschirm) wird ein neues Objekt geschaffen mit Operationen, die das NDBS zur Verfügung stellt. Anschließend wird es dann wieder mit den GDSS-Operationen ausgegeben.

Steht ein solches GDSS zur Verfügung, so bleiben dem Entwickler eines speziellen CAD-Systems nur noch folgende Aufgaben:

- Auswahl und Anschluß der graphischen Geräte; Definition in der Arbeitsplatzbeschreibungstabelle von GKS (Konfigurierung von GKS),

- Definition des Datenbankschemas aus komplexen Objekten, die sich aus räumlichen Objekten und nichträumlichen Attributen (Bezeichnung, Material, Gewicht, Hersteller, Preis, ...) zusammensetzen, sowie aus einfachen Objekten (Kataloge, Tabellen usw.),
- Programmierung anwendungsspezifischer Verfahren zur Verknüpfung von räumlichen und nichträumlichen Informationen, z.B. Simulationen zur Prüfung von Tragfähigkeit und Biegefestigkeit, Belastungstests, Bearbeitungsverfahren, Erzeugung von NC-Programmen usw.,
- Programmierung des Benutzerdialogs, Verwaltung der Bildschirmaufteilung, Kommandointerpretation.

Aus den soweit beschriebenen Anforderungen läßt sich erkennen, daß die Implementierung eines solchen GDSS sicher nicht sehr einfach ist, wenn man den gewünschten Grad an Allgemeinheit erreichen will. Deshalb soll nun noch eine abgeschwächte Lösung diskutiert werden, die nicht den vollen Funktionsumfang bietet, dafür aber einfacher zu realisieren ist und es so eher erlaubt, die Tauglichkeit des Konzepts der Integration von NDBS und GKS in der Praxis an einer Art Prototyp zu erproben.

3.3 Enge Kopplung von graphischem DBS und GKS

Bei dieser abgeschwächten Lösung der engen Kopplung des NDBS mit GKS wird auf die bei der Integration eingeführte zusätzliche Abbildungsschicht (GDSS) verzichtet. Dafür wird eine direkte Schnittstelle zwischen dem NDBS und GKS eingeführt, die einen Aufruf der GKS-Funktionen durch das NDBS erlaubt (Abb. 6). Um diese Schnittstelle ausnutzen zu können, wird zusätzlich der Funktionsumfang des NDBS zum AP hin erweitert. Diese Erweiterung ermöglicht dem AP die direkte graphische Darstellung eines DB-Objekts durch das NDBS. Es ist dem AP bekannt, daß aus seinem Aufruf an das NDBS zahlreiche Aufrufe an GKS resultieren können und GKS sich demnach anschließend in einem anderen Zustand befindet. Aus der Sicht des AP bilden GKS und das NDBS also zwei getrennte Komponenten mit dem jeweils üblichen Funktionsumfang, die jedoch zusätzlich über die angesprochene Schnittstelle gekoppelt sind. Die Zusammenarbeit dieser beiden Komponenten, die bei der Integration durch das GDSS geregelt und verwaltet wurde, muß nun teilweise wieder vom AP koordiniert werden.

Der Entwickler eines speziellen CAD-Systems muß also neben den Algorithmen für die bereits bei der Integration angesprochenen Aufgaben (siehe unter 3.2) auch Algorithmen für diese Aufgabe bereitstellen. Diese Algorithmen sind jedoch im allgemeinen wesentlich einfacher als die entsprechenden Algorithmen des GDSS, da sie auf die speziellen Bedürfnisse des zu entwickelnden CAD-Systems zugeschnitten werden können, während die Algorithmen des GDSS einen gewissen Grad an Allgemeinheit erreichen müssen.

Die wichtigste Aufgabe für das AP ist dabei die Konsistenzüberwachung zwischen der graphischen Darstellung auf dem Ausgabegerät und der Darstellung in der DB und, damit verbunden, die Umsetzung einer graphischen Darstellung in die zugehörige DB-Darstellung. Die Umsetzung in die andere Richtung, also von der DB-Darstellung in die gewünschte graphische Darstellung, wird ja durch das NDBS vorgenommen.

Nachfolgend soll eine einfache Möglichkeit zur Lösung dieser Aufgabe vorgestellt werden:

- Ausgabe eines gespeicherten Objekts
  Wird durch den Benutzer lediglich die graphische Darstellung eines bereits gespeicherten Objekts verlangt, kann diese Anforderung vom AP direkt an das NDBS weitergereicht werden. Dabei ist es sogar unnötig, weitere Daten, d.h. die räumliche und die nichträumliche Information, an das AP zurückzuliefern.

- Modifikation eines gespeicherten Objekts
  Anders sieht es dagegen bei der Modifikation eines bereits gespeicherten Objekts aus. Dazu wird zunächst durch das AP die Ausgabe der graphischen Darstellung durch das NDBS veranlaßt. Gleichzeitig mit dieser Ausgabe müssen die räumlichen Informationen, z.B. der CSG-Operationsbaum mit Primitivkörpern als Blattknoten (/Re80/), und die nichträumlichen Zusatzinformationen dem AP zur Verfügung gestellt werden. Der Benutzer ändert nun das am Bildschirm dargestellte Objekt, beispielsweise subtrahiert er an einer bestimmten Stelle den Primitivkörper Zylinder, um eine zusätzliche Bohrung anzubringen. Es ist Aufgabe des AP, diese Änderung auch in der DB zu vermerken.

Versäumt es dies aufgrund eines Fehlers in der Programmierung, gehen die Änderungen ganz oder teilweise verloren, da es keine übergreifende und konsistenzüberwachende Instanz wie das GDSS gibt.

- Eingabe eines neuen Objekts
  Entweder ist es Aufgabe des AP, aus zweidimensionalen Ansichten (technische Zeichnung) die dreidimensionale Form abzuleiten (/SG83/) und in der Datenbank zu speichern, oder der Konstrukteur arbeitet direkt mit den dreidimensionalen Primitivkörpern des CSG-Modells, was wesentlich einfacher erscheint. Das AP kann seine Anweisungen dann nahezu unverändert an das NDBS weiterreichen (vgl. unter 2.1).

Diese Lösung hat, wie erwähnt, den Nachteil, daß die Programmierung des AP mit zusätzlichen Aufgaben belastet ist und eine potentielle Fehlerquelle darstellt. Sie kann jedoch als Ausgangspunkt für weitere Entwicklungen betrachtet werden, die u.U. bis zur Integration von GKS und einem NDBS führen.

4. Zusammenfassung

Zusammenfassend kann gesagt werden, daß auf dem Gebiet der DB-Unterstützung für CAD-Arbeitsplatzsysteme noch einige Probleme zu lösen sind. Wesentliche Überlegungen betreffen dabei die Rechnerkonfiguration eines derartigen Arbeitsplatzsystems, das zu verwendende DBS und die Zusammenarbeit des DBS mit dem CAD-System. Dieser Artikel beschäftigte sich hauptsächlich mit dem letzten Punkt. Als Grundlage wurden dazu ein spezielles, auf die graphische Datenverarbeitung und ihre Aufgaben zugeschnittenes DBS, ein NDBS, und eine geeignete Rechnerkonfiguration, das Database-Front-End, beschrieben. Danach wurden drei Ansätze für die Zusammenarbeit eines DBS (oder NDBS) mit einem CAD-System vorgestellt. Die beiden ersten Ansätze, die unter dem Begriff der losen Kopplung zusammengefaßt werden können, sind dabei ad-hoc-Lösungen, die keine oder nur geringfügige Änderungen an den beiden Systemen verlangen. Der dritte Ansatz, die Einbettung des DBS oder besser des NDBS in das CAD-System, macht dagegen verschiedene Änderungen erforderlich, die jedoch von der Art der Einbettung abhängig sind. Mit der Integration und der engen Kopplung des NDBS mit GKS wurden zwei Möglichkeiten der Einbettung

vorgestellt. Dabei liegt es nahe, zunächst die enge Kopplung als Prototyp zu implementieren und daran Erfahrungen für eine Weiterentwicklung in Richtung auf die integrierte Lösung hin zu sammeln.

Literaturverzeichnis:

/CAD84/ o.A.: CAD und Datenbanksysteme, computer magazin, Vol. 13, Nr. 5, Mai 1984, S. 16 - 20

/Ch84/ Christmann, H.-P.: Rechnerarchitekturen zur Unterstützung nichtkommerzieller Anwendungen, Forschungsbericht SFB 124 Nr. 08/84, Universität Kaiserslautern, Mai 1984

/Da81/ Date, C.J.: An Introduction to Database Systems, 3rd Edition, Addison-Wesley Publ. Co., Reading, Mass., 1981

/En84/ Enderle, G.: IGES (Das aktuelle Schlagwort), Informatik-Spektrum, Band 7, Heft 1, Febr. 1984, S. 45

/GKS83/ Das Graphische Kernsystem GKS, mehrere Artikel in: Informatik-Spektrum, Band 6, Heft 2, April 1983

/HR83/ Härder, T., Reuter, A.: Database Systems for Non-Standard-Applications, in: H. J. Schneider (ed.): Proc. ICS 83 Nürnberg, German Chapter of the ACM, Bericht Nr. 13, Teubner Verlag, Stuttgart 1983, S. 452 - 466

/Mi84/ Mitschang, B.: Überlegungen zur Architektur von Datenbanksystemen für Ingenieuranwendungen, erscheint im Tagungsband der GI-Jahrestagung 1984 in Braunschweig

/Re80/ Requicha, A.: Representation of Rigid Solid Objects, in: J. Encarnacao (ed.): Computer Aided Design, Springer-Verlag, Lecture Notes on Computer Science 89, New York 1980, S. 1 - 78

/SG83/ Sakurai, H., Gossard, D.C.: Solid Model Input Through Orthographic Views, in: Tanner, P. (ed.): SIGGRAPH '83 Conf. Proc., Computer Graphics, Vol. 17, No. 3, July 1983, S. 243 - 252

/Si84/ Sikeler, A.: Untersuchung von Speicherungsstrukturen für dreidimensionale Objekte (Arbeitstitel), Forschungsbericht, Universität Kaiserslautern, Fachbereich Informatik (in Vorb.)

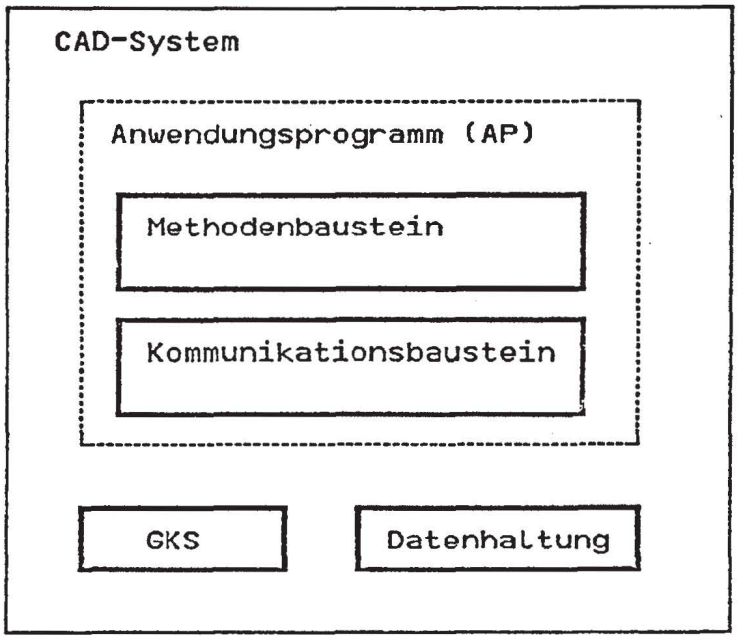

Abb. 1: Struktur eines CAD-Systems nach Anderl et al. (/GKS83/)

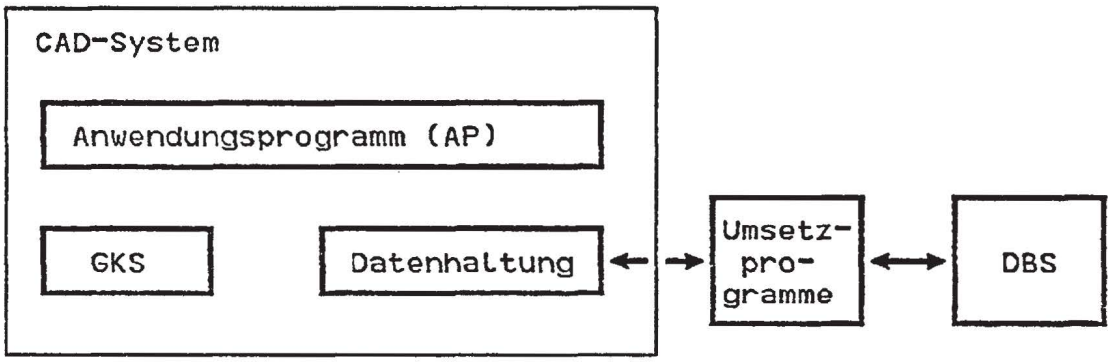

Abb. 2: Lose Kopplung an ein geschlossenes CAD-System

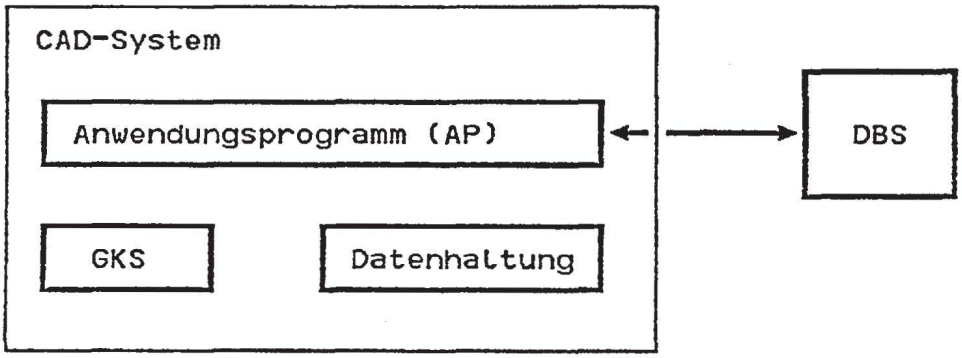

Abb. 3: Lose Kopplung an ein offenes CAD-System

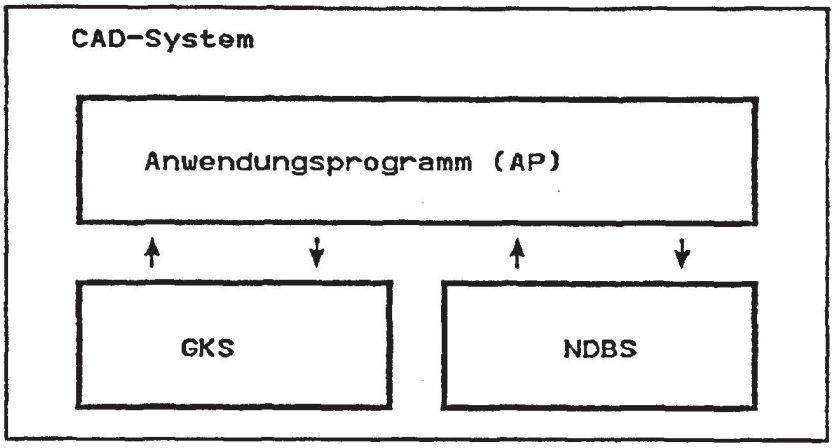

Abb. 4: Einbettung des DBS in ein CAD-System

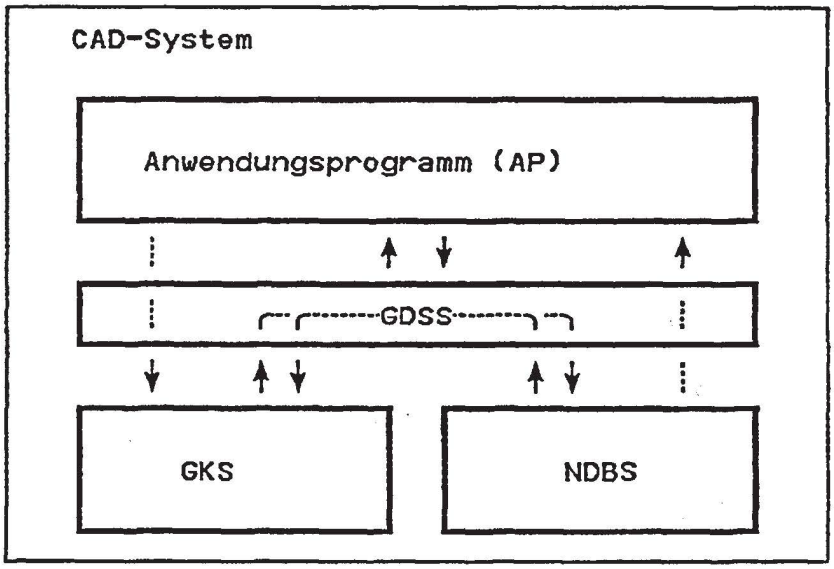

Abb. 5: Integration von graphischem System und NDBS durch das Graphische Darstellungs- und Speicherungssystem (GDSS)

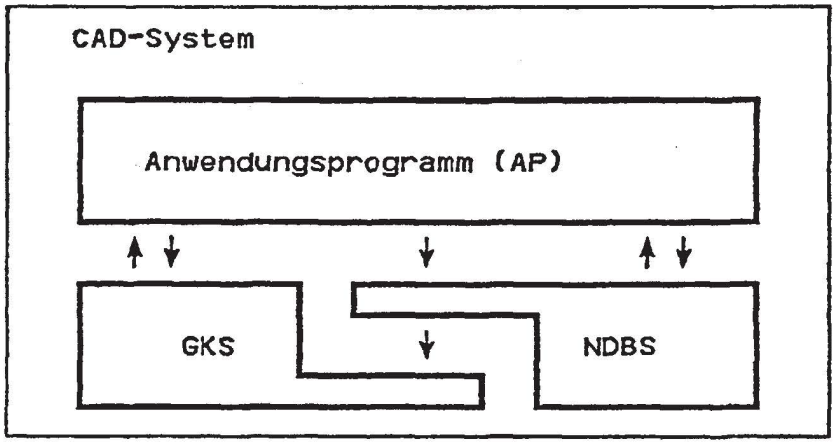

Abb. 6: Enge Kopplung von graphischem System und NDBS