Thomas Krebs

Integration elektromechanischer CA-Anwendungssysteme über einem STEP-Produktmodell

Thomas Krebs

Integration elektromechanischer CA-Anwendungssysteme über einem STEP-Produktmodell

Herausgegeben von
Professor Dr.-Ing. Klaus Feldmann,
Lehrstuhl für
Fertigungsautomatisierung und Produktionssystematik

FAPS

Meisenbach Verlag Bamberg

Als Dissertation genehmigt von der Technischen Fakultät
der Friedrich-Alexander-Universität Erlangen-Nürnberg

Tag der Einreichung:	3. Mai 1996
Tag der Promotion:	26. Juli 1996
Dekan:	Prof. Dr.-Ing. Stoyan
Berichterstatter:	Prof. Dr.-Ing. K. Feldmann
	Prof. Dr. H. Wedekind

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Krebs, Thomas:
Integration elektromechanischer CA-Anwendungssysteme über einem STEP-Produktmodell / Thomas Krebs. Hrsg. von Klaus Feldmann. - Bamberg : Meisenbach, 1997
(Fertigungstechnik - Erlangen ; 59)
Zugl.: Erlangen, Nürnberg, Univ., Diss., 1995
ISBN 3-87525-081-8 ISSN 1431-6226
NE: GT

Herstellung: Gruner Druck GmbH, Erlangen-Eltersdorf
Printed in Germany

Vorwort

Die vorliegende Dissertation entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Fertigungsautomatisierung und Produktionssystematik der Friedrich-Alexander-Universität Erlangen-Nürnberg

Herrn Professor Dr.-Ing K. Feldmann, Leiter des Lehrstuhls für Fertigungsautomatisierung und Produktionssystematik, danke ich herzlich für die engagierte Förderung meiner Arbeit sowie für den Freiraum, den er mir bei der Bearbeitung dieses wissenschaftlichen Themas ließ. Ebenso bedanke ich mich bei Herrn Professor H. Wedekind, Leiter des Lehrstuhls für Datenbanken, für die Übernahme des Koreferats sowie für die lebendigen Diskussionen, die während der Bearbeitung dieser Arbeit stattfanden.

Ferner gilt mein Dank allen Kollegen, Studenten und wissenschaftlichen Hilfskräften, die durch ihre Unterstützung zum Gelingen dieser Arbeit beigetragen haben. Hierbei möchte ich mich besonders bei Herrn D. Janetzko für die Unterstützung bei der Erstellung der Grafiken, Herrn E. Siegel für seinen Einsatz bei der Systementwicklung sowie bei Herrn J. Leiblein für interessante Diskussionen bedanken.

Vielen Dank möchte ich allen Kollegen zukommen lassen, die im Rahmen des ISO-STEP-Projektes gearbeitet haben und durch engagierte Diskussion und ständige Anregung das Gelingen dieser Arbeit gefördert haben.

August 1996

Thomas Krebs

Integration elektromechanischer CA-Anwendungen über einem STEP-Produktmodell

Inhaltsverzeichnis

1 Einleitung

Die Fähigkeit, Güter in hoher Stückzahl unter vorgegebener Qualität bei niedrigen Kosten herstellen zu können, ist in großem Maße an der Spezialisierung einzelner Leistungserbringer auf spezifische Aufgabenbereiche innerhalb einer industriellen Wirtschaftsgemeinschaft gebunden /1/. Die Arbeitsteilung ermöglicht durch die Erlangung höherer Fertigkeiten im Produktionsprozeß eine höhere Effizienz der einzelnen Produzenten. Die Arbeitsteilung in hochentwickelten Industrienationen spiegelt sich in der Strukturierung der Wirtschaft in einzelne Teilnehmer als auch in der Aufbauorganisation der einzelnen Wirtschaftsobjekte in Unternehmen, Abteilungen und Gruppen.

Ein wesentliches Element der Kommunikation zwischen einzelnen Produktionspartnern in der industriellen Fertigung war lange Zeit die Technische Zeichnung. Gab es Skizzen technischer Gebilde bereits im Altertum, so bei Römern und Griechen, als auch im Mittelalter, etwa bei Michelangelo, so entstand die Technische Zeichnung, wie sie auch heute noch bekannt ist, im England der industriellen Revolution. Kennzeichnend für die Nutzung der technischen Zeichnung war die Trennung der planenden Tätigkeit für ein Technisches Gebilde, die damals in der Hand der Meisters lag und der ausführenden Tätigkeit durch den Maschinisten.

Mit der Entwicklung des elektronischen Rechners und der daraus resultierenden numerisch gesteuerten Werkzeugmaschine 1951 am MIT setzte eine Entwicklung ein, die heute allgemein als zweite industrielle Revolution bezeichnet wird. Für die Fertigungstechnik bestand nun die Möglichkeit, im Schatten der kommerziellen Computeranwendungen einen Produktivitätssprung zu erreichen, der dem Ersatz der manuellen Fertigung durch die maschinelle gleichkommt.

Die Vorgabe der Steueranweisung für einen numerisch gesteuerten Automaten induzierte die Entwicklung von computergestützten Werkzeugen zur elektronisch basierten Definition des zu fertigenden Bauteils. Mit Hilfe dieser abstrahierten Produktinformation wurde die Fertigungsinformation abgeleitet, und dem Bearbeitungsautomaten zur Verfügung gestellt.

Für die diversen Produktbereiche, exemplarisch sind der Maschinenbau mit seiner geometrisch orientierten Beschreibung und die Elektrotechnik oder Elektronik mit ihrer logisch- bzw. verbindungsorientierten Sicht des Produktes genannt, haben sich unterschiedliche Beschreibungsmodelle für die zu produzierenden Objekte etabliert. Die in speziellen Anwendungssituationen nötigen unterschiedlichen Produktinformationen haben daraus zu einer beachtlichen Anzahl von spezialisierten Anwendungssystemen zur Beschreibung des Produktionsziels, des Produktes, geführt. Hersteller und Kunde haben jedoch nach wie vor den Wunsch, Information über das Objekt ihrer Beziehung in einem wie vormals verständlichen Rahmen auszutauschen.

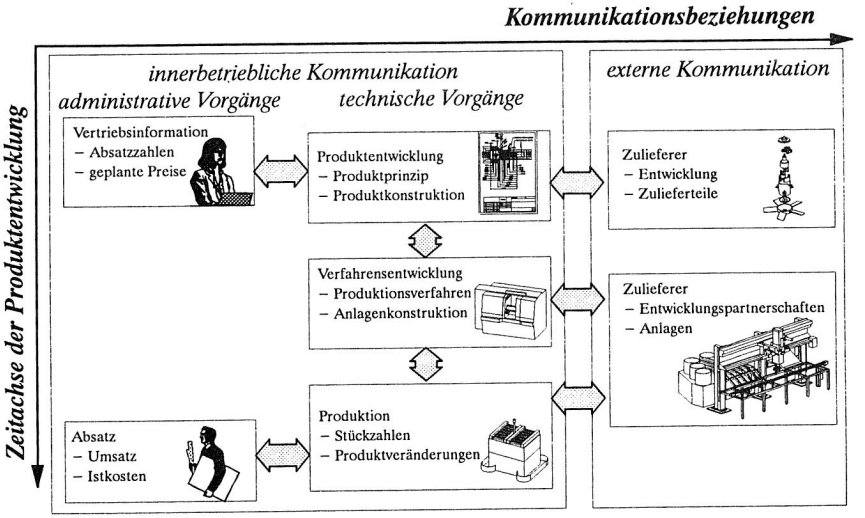


Abb. 1 Innerbetriebliche und externe Kommunikation durch Produktdatenaustausch

Über die Beziehung zwischen Zulieferer-Kunde hinaus besteht auch ein Kommunikationsproblem innerhalb einer wirtschaftlichen Einheit, wie sie ein Unternehmen darstellt (siehe Abb. 1). Innerhalb der Technischen Auftragsabwicklung werden in unterschiedlichen, nicht notwendigerweise sequentiellen Arbeitsgängen Informationen über das Produkt erzeugt, verändert oder erweitert. Für die Gesamtheit aller Informationen, die das Produkt umfaßt, hat sich der Begriff des Produktmodells etabliert.

1.1 Problemdefinition

Kritische Erfolgsfaktoren für produzierende Unternehmen unterscheiden sich in verschiedenen Industriebereichen deutlich voneinander. Sind für Produkte elektronischer Basis hohe Innovationsraten bei Bauteilen und Produktionstechnologie, Miniaturisierung und starker Preisdruck kennzeichnend, so kann bei mechanischen Produkten die wesentliche Problematik in der logistischen Beherrschung der Variantenvielfalt, die oftmals eine Geometrievarianz bedeutet, identifiziert werden.

Werden in einem Produkt wesentliche Funktionen sowohl durch mechanische als auch durch elektronische Komponenten sichergestellt, so können sich die Kriterien elektronischer und mechanischer Entwicklung/Konstruktion und Fertigung miteinander vermengen. Für Produkte derartigen Aufbaus wird oftmals der Begriff Mechatronik verwendet. Die Anzahl der heute existierenden Produkte, die sowohl elektronische als auch mecha-

nische Komponenten enthalten ist jedoch derart groß[1], daß eine Definition des Begriffs Mechatronik in dem hier gebrauchten Sinne angebracht erscheint /24/:

Mechatronik beschreibt Produkte bzw. Verfahren zur Entwicklung, Konstruktion und Fertigung von Produkten die sowohl elektronischen als auch mechanischen Funktionsanteil besitzen, bei denen eine direkte funktionsrelevante Abhängigkeit elektronischer funktionaler Eigenschaften mit mechanischen funktionalen Eigenschaften besteht.

Die Notwendigkeit des Einsatzes computerunterstützter Entwurfswerkzeuge für elektronische Anwendungen (EDA, ECAD, CAE) als auch Konstruktionshilfsmittel im mechanischen Anwendungsbereich (CAD, MCAD) ist oftmals erläutert worden /95/. Für mechatronische Produkte gilt dies in speziellem Maße.

1.2 Aufgabenstellung

Mit der vorliegenden Arbeit soll ein wesentlicher Beitrag erbracht werden, Grundlagen für eine integrierte Ingenieurarbeitsumgebung für den Entwurf mechatronischer Produkte bereitzustellen. Der stark strapazierte Begriff der Integration wird hier in dem Sinne benutzt, die getrennten Welten der Ingenieurtätigkeiten des Maschinenbaus mit der Ingenieurwelt der Elektrotechnik zu vereinen. Der anwendungsneutrale Ansatz von STEP bietet hierzu erstmals eine ernstzunehmende Grundlage. Ohne diese Grundlagen wäre ein zukunftsorientierter Ansatz in dem existierenden Umfeld nicht möglich und auch nicht sinnvoll gewesen.

1.3 Überblick über die Arbeit

Die Darstellung der Arbeit ist mit einer ausführlichen Beschreibung der benutzten Prinzipien, Normen und Systeme verbunden. Dies liegt in der Tatsache begründet, daß in wesentlichen Bereichen eine im ingenieurwissenschaftlichen Sinne kompositorische Arbeit vorliegt. Die weitgehende Einbeziehung existierender Paradigmen und Systeme stellt insoweit eine Vision für einen schrittweisen Übergang von partikulären, geschlossenen Systemen zu einem Satz modulartiger, offener und konfigurierbaren Baukastensysteme dar.

Die Beschreibung der zugrundeliegenden Prinzipien soll im wesentlichen die Problematik der Realisierbarkeit unterstreichen und wird im Hinblick auf diese Problematik betrieben.

---

1. *Prinzipiell hat jedes Produkt mit elektronischer Funktionalität einen mechanischen Aufbau.*

Als der wesentliche Baustein der Arbeit dient die ISO10303, die in nötiger Ausführlichkeit in Kapitel 2 dargestellt wird. Hierbei muß auf die Kerntechnologie, die Modellierungssprache EXPRESS besonders eingegangen werden. In vielen Veröffentlichungen wird im Zusammenhang mit der ISO10303 oftmals von EXPRESS/STEP gesprochen, da diese Modellierungssprache einen derart wesentlichen Anteil am Aufbau, Verständnis und der Realisierbarkeit der gesamten Norm hat, daß diese wohl als eine Basistechnologie von STEP angesehen werden kann, wie in Kapitel 2 deutlich werden wird. Die in der Norm ISO-10303 spezifizierten Implementierungstechniken werden vorgestellt. Der für das weitere Verständnis fundamentale Ansatz des SDAI (Standard Data Access Interface) wird dabei nötiger Raum geboten.

Kapitel 3 zeigt die verschiedenen STEP-Realisierungen, die zum Entstehungszeitpunkt der Arbeit bekannt waren auf und vergleicht deren technologischen Ansatz. Als wesentlicher Schwerpunkt wird dabei die Realisierung der Abbildung der Modellierungssprache EXPRESS auf ein System in den jeweiligen Laufzeitumgebungen bilden. Die für die Sprache EXPRESS vorhandenen Compiler bilden dabei die technologischen Grundlagen für jegliche Realisierungen. Ein kurzer Überblick über den Stand der Technologie von STEP-Prozessoren zeigt erste industriell verfügbare STEP-Realisierungen. Das im Rahmen der Arbeit entstandene Entwicklungswerkzeug für STEP-Anwendungen kann hierbei für alle normierten Implementierungstechniken benutzt werden.

Die Realisierung von STEP-kompatiblen Prozessoren, die in Kapitel 3 vorgestellt werden, sind nur ein Zwischenschritt zu Systemen, die eine STEP-orientierte Integration von technischen Anwendungssystemen vollziehen werden. Durch die Möglichkeit, technische Softwaresysteme über SDAIs, wie in Kapitel 3 gezeigt, zu integrieren, bietet sich die Chance, existierende Anwendungssysteme von ihren proprietären Mechanismen zur Nutzung von Hintergrundspeicher zu lösen und eine Integration dieser Systeme im Sinne einer einheitlichen Datenverwaltung zuzuführen.

Datenbanken haben im Bereich kommerzieller Anwendungen einen hohen Stellenwert durch die Bereitstellung einer zentralisierten, standardisierten und sicheren Integrationsquelle erreicht. Die Bemühungen diesen Keimzelle einer zumindest bereichsweisen Integration auch technischen Anwendungen zukommen zu lassen, scheiterte bisher an den unterschiedlichen Modellierungsparadigmen zwischen den kommerziellen Datenbankmanagementsystemen und den komplexen technischen Anwendungssystemen. STEP bietet durch seinen methodischen Modellierungsansatz prinzipiell die Möglichkeit, eine Abbildung technischer Modelle auf beliebige Datenhaltungssysteme zu realisieren. In Kapitel 4 werden unter dem Ansatz einer Abbildung von EXPRESS auf Datenbanken die unterschiedlichen Prinzipien von Datenbankmanagementsystemen daraufhin untersucht, ob deren Einsatz eine sinnvolle Nutzung als STEP-Produktdatenbank gewährleisten kann.

Als prototypische Realisierung einer STEP-Produktdatenbank wird die Abbildung von EXPRESS auf das Erweitert Relationale Datenbankmanagementsystem Postgres dar-

gestellt und diskutiert. Die maschinelle Generierung der Datenbankschemata für Postgres ist dabei Teil der Entwicklungsumgebung für STEP-Anwendungen.

Die Verarbeitung von technischen Produktinformationen wird vornehmlich in Konstruktionssystemen (CAD) vorgenommen. Die Bereitstellung von STEP-Produktdatenbanken erlaubt zunächst den Zugriff auf diese Informationen mit den der Datenbank inhärenten Anfragemechanismen, z.B. ad-hoc Anfragesprachen. Den sinnvollen Umgang mit Produktinformationen können jedoch nur CAD-Systeme bieten, die in der Lage sind eine STEP-Produktdatenbank als Speichermedium zu nutzen.

In Kapitel 5 wird die Anbindung eines geometrischen Modellierungssystems an eine STEP-Produktdatenbasis erläutert. Die hierfür nötigen Grundlagen der Repräsentation technischer Objekte wird geschaffen. Als ein Beispiel dieser Integration wird die Anbindung von CAD-Systemen anhand der Kopplung eines Geometriemodellierers gezeigt. Die von dieser Anbindung gestellten Anforderungen an das Modellierungssystems wird aufgezeigt und führt zu der Auswahl des verwendeten Systems.

Während die in den Kapiteln 3 bis 5 dargestellten und realisierten Problemlösungsmethoden weitgehend unabhängig von bestimmten branchenspezifischen Anwendungsproblemen genutzt werden können, wird die Anwendungsorientierung der STEP Anwendungsprotokolle eingeführt. Die Problematik der Modellbildung für spezifische Anwendungsbereiche führt zur Integrationsproblematik über diese Anwendungsprotokolle hinweg. Die Problemstellung wird identifiziert und Lösungsvorschläge dargestellt.

Die in den Kapiteln 3 bis 5 bereitgestellte Umgebung für STEP-Anwendungen wird benutzt um einen Prototyp eines integrierten Entwurfssystems für mechatronische Anwendungen zu realisieren. Die Anforderung aus einem neuem Produkttypus führt zum Bedarf an ein derartiges System. Die Produkt- und Fertigungstechnologie der zugrundeliegenden Entwicklung wird soweit es für das Verständnis des Entwurfssystems nötig ist, dargestellt. Die Anforderungen werden verdeutlicht und bilden das Pflichtenheft für den Prototypen. Dessen Realisierung innerhalb der STEP-Entwicklungsumgebung wird verdeutlicht und ein Anwendungsfall beschrieben.

2 STEP als Basis integrierter Produktmodelle

Der Bedarf, Informationsaustausch zwischen Anwendungssystemen im Konstruktions- und Entwicklungsbereich vornehmen zu können, ist durch die in marktwirtschaftlichen Industrien vorzufindende Vielfalt solcher Anwendungssysteme bei Kunde und Lieferant begründet.
Der Wunsch nach einheitlichen Datenaustauschformaten zwischen diesen Systemen und ein damit einhergehender Normierungsbedarf darf jedoch nicht einer Anwendungsvielfalt entgegenwirken. Eine Normierung darf den technischen Fortschritt nicht behindern.
Die bisherigen, oftmals nationalen Normierungsbemühungen münden seit etwa 1984 in eine von der ISO (International Standardization Organization) zu normierende Entwicklung ein.

Die in der ISO 10303 zur Normung anstehende Entwicklung STEP (**St**andard for the **E**xchange of **P**roduct Model Data) löst mit ihrer Erscheinung alle Datenaustauschformate in CA-Bereichen ab und stellt darüberhinaus eine Entwicklungsmethodik für CA-Anwendungssysteme bereit.
Das vorliegende Kapitel soll das für das Verständnis der weiteren Arbeit nötige Grundwissen bereitstellen.

Zunächst soll durch die Darstellung der Entwicklung der Datenaustauschformate die Einordnung von STEP ermöglicht werden. Danach folgt eine kurze Einführung in die Teile der Norm, die ob ihrer Komplexität in eine offene Zahl von Teilen zerlegt wurde. Die mit STEP eingeführten Prinzipien und Methoden sollen letztendlich einer kritischen Überprüfung unterzogen werden.

2.1 Defizite vorliegender Produktmodelle führen zu STEP

Die Entwicklung von STEP war aus der Erkenntnis entstanden, daß die bisher zur Verfügung stehenden Datenaustauschmöglichkeiten den Anforderungen industrieller Anwendung nicht genügen. Defizite der vor STEP zur Verfügung stehenden Formate waren begrenzte Anwendbarkeit, mangelnde Erweiterbarkeit, schlechte Übertragungsqualität und anbieterspezifische Interpretationen der Normen durch mißverständliche Dokumentation.

2.1.1 Strategien für den Datenaustausch

Der Bedarf Daten aus verschiedenen CAD-Systemen auszutauschen entstand mit der steigenden Vielfalt industriell eingesetzter Systeme. Für den reinen Austausch lassen sich prinzipiell zwei verschiedene Strategien skizzieren:

1. Direkter Austausch über eigene Prozessoren.

2. Ein neutrales Zwischenformat mit Prä- und Postprozessoren für jedes System.

Während Lösung 1 meist die höhere Übertragungsleistung für genau die zugrundeliegende Systemkombination verspricht, muß aus ökonomischen Gründen jedoch Lösung 2 favorisiert werden. Für den direkten Austausch sind für n Systeme insgesamt n(n − 1) Prozessoren nötig, bei Nutzung eines neutralen Formates jedoch nur 2n. Daher ergibt sich ab n = 3 eine rechnerisch günstigere Situation bei einem neutralen Format. Bei etwa 150 industriell eingesetzten CAD-Systemen weltweit ist die zweite Lösung die einzig praktikable, um einen Austausch zwischen allen Systemen zu ermöglichen.

2.1.2 Historie der Entwicklungen von neutralen Austauschformaten

Die oben angesprochene Problematik wurde früh erkannt, und so entstanden eine Reihe von neutralen Austauschformaten, die jedoch nur teilweise industrielle Bedeutung erlangten. In /83/ wird versucht, eine historische Entwicklungslinie der neutralen Austauschformate aufzuzeigen. Dabei wird sowohl auf Austauschformate für Mechanik-CAD-System als auch für Elektronik-CAD-Systeme eingegangen.

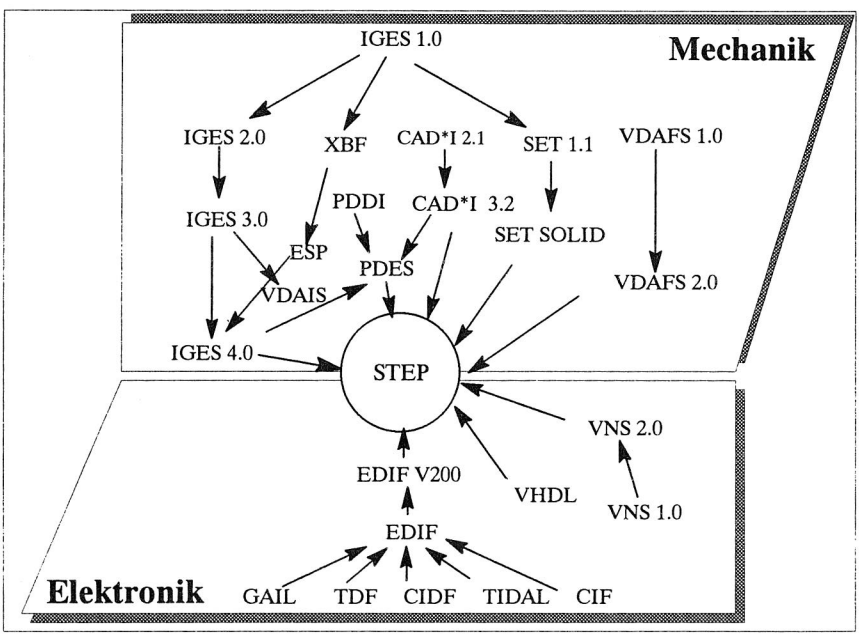


Abb. 2 Entwicklungsgeschichte der neutralen CAD-Austauschformate

Von den dargestellten Formaten haben im wesentlichen nur IGES, VDA-FS, SET[2] und VHDL industriell größere Bedeutung erlangt. Viele der vorgestellten Formate sind nur zu Forschungszwecken an unabhängigen Instituten entwickelt worden und sind von den Systemanbietern nicht aufgenommen worden. Bei den gezeigten Entwicklungen handelt es sich ausschließlich um Schnittstellenspezifikationen, die eine persistente Dateiebene besitzen. Schnittstellen, die primär als Grafik-APIs wie GKS oder PHIGS entwikkelt wurden sind hier nicht von Interesse.

Von den in Abb. 2 genannten Spezifikationen sind vor allem die Entwicklungen PDES, CAD*I und EDIF zu nennen, bei deren Entstehung viel Wissen für die Entwicklung von STEP gewonnen wurde.

Wesentliches Defizit der meisten Entwicklungen aus Abb. 2 war das Fehlen einer formalen Entwicklungsmethode sowie die Dokumentation der Entwicklung, die für das Verständnis des neutralen Formates wesentlich ist.
Die Unterteilung der beiden Haupteinflüsse für die STEP-Entwicklung aus der Mechanik und der Elektronik ist hier nur von beispielhaftem Interesse. Es zeigt sich jedoch, daß bisher keine Entwicklungen die Berührungspunkte zwischen Mechanik und Elektronik unterstützen.

2.1.3 Anforderungen an die STEP-Entwicklung

Im Juli 1984 wurden durch die Vertreter der nationalen Normierungsgremien die ISO TC184/SC4 (International Standardization Organization Technical Committee 184 Subcommittee 4) gebildet, um einen einzigen internationalen Standard für den Produktdatenaustausch zu definieren. Um die existierenden Modelle bzw. Formate sinnvoll in STEP zu integrieren, sollte für die Entwicklungszeit von STEP versucht werden, eine Kompatibilität der existierenden Formate zu STEP herzustellen. Dies sollte durch die Nutzung gemeinsamer Referenzmodelle ermöglicht werden.

Die Entwicklungziele von STEP wie sie zu Beginn der Definition aufgestellt wurden, lassen sich sich wie folgt zusammenfassen /4/:

- Vollständigkeit: STEP soll die vollständige Beschreibung eines Produktes ermöglichen und dabei sowohl den Datenaustausch als auch die Archivierung unterstützen.
- Erweiterbarkeit: Trotz der Allgemeingültigkeit von STEP soll ein Rahmen bereitgestellt werden, um Erweiterungen vornehmen zu können.
- Testfähigkeit der Erweiterungen: Erweiterungen des Standards sollen einer Begutachtung identisch dem Standard unterworfen werden, bevor Implementierungen vorgenommen werden.

2. *SET hat nur innerhalb Frankreichs signifikante Bedeutung*

- Effizienz: STEP soll effizient hinsichtlich der Rechnerressourcen wie CPU-Belastung und Speichergröße sein.
- Kompatibilität mit anderen Standards: Um die Migration von anderen Standards zu erleichtern soll STEP weitgehend mit den relevanten Standards kompatibel sein. Es soll soweit möglich andere Standards nutzen wie z.B.: Grafik, Zeichensätze und Terminologie.
- Minimale Redundanz: Es soll nur eine Möglichkeit geben, ein bestimmtes Konzept zu beschreiben.
- Rechnerunabhängigkeit: STEP soll unabhängig von Hardware und Basissoftware sein.
- Logische Klassifikation von Datenelementen: STEP soll Untermengen für die Implementierung definieren.
- Ein Rahmen zur Konformitätsüberprüfung soll Bestandteil der Norm sein, um Konformität von Implementierungen normgerecht vornehmen zu können.

Retrospektiv kann gesagt werden, daß die Anforderungen an STEP, soweit sie zum gegenwärtigen Zeitpunkt abschließend beurteilt werden können, von den vorliegenden Teilen der Norm im wesentlichen erfüllt werden. Dabei ist allerdings auch zu beobachten, daß einige Anforderungen abgeschwächt wurden, wie z.B. die Forderung nach Effizienz. Dies ist mit der überproportionalen Verbesserung der Rechnerleistung seit der STEP-Definitionsphase zu begründen.

2.2 Aufbau der Norm ISO10303 – STEP

Die im Laufe der Entwicklung an Umfang und Komplexität zunehmende Entwicklung STEP wurde etwa 1986 in einzelne Teile aufgespalten, um diese einerseits in ihrer Komplexität erfassen zu können und andererseits die Möglichkeit zu schaffen, eine modulare Entwicklungsmethodik zu schaffen.

Die einzelnen Teile der Norm ISO 10303 werden in einer aufsteigenden Zählung identifiziert. Teile der Norm, die in einem größeren methodischen Zusammenhang stehen, sind in sogenannten Reihen zusammengefaßt. Für die einzelnen Teile wurden Bereichsverantwortliche, Editoren für das Normdokument und Mitglieder der Arbeitsgruppen ernannt.

Zu Beginn einer Entwicklung eines Teils der Norm wird ein Zeitplan zur Entwicklung vorgelegt. In fest definierten Abständen werden die Ergebnisse der Arbeitsgruppe vor den Mitgliedern der ISO vorgestellt. Dabei können die ISO-Vertreter ihre Vorschläge oder Einwände einbringen. Zu bestimmten Meilensteinen erreicht der Teil vorgegebene Stati, die halboffiziellen Charakter darstellen. Zum Zeitpunkt der Verabschiedung vor der ISO wird das offizielle Dokument vorgestellt. Die stimmberechtigten Mitglieder der ISO sind

dann berechtigt ihre Zustimmung oder Ablehnung zu den einzelnen Teilen abzugeben. Im Falle der Ablehnung besteht die Möglichkeit einer Revision. Sollte diese ebenfalls abgelehnt werden, ist der zur Abstimmung stehende Teil der Norm abgelehnt. Sollte trotzdem Interesse an diesem Teil der Norm bestehen, muß der Entwicklungsvorgang wieder aufgenommen werden.

Wie oben erläutert wurden die Teile der ISO 10303 methodisch verwandten Reihen zugeordnet. Diese sind wie folgt (Abb. 3) strukturiert:

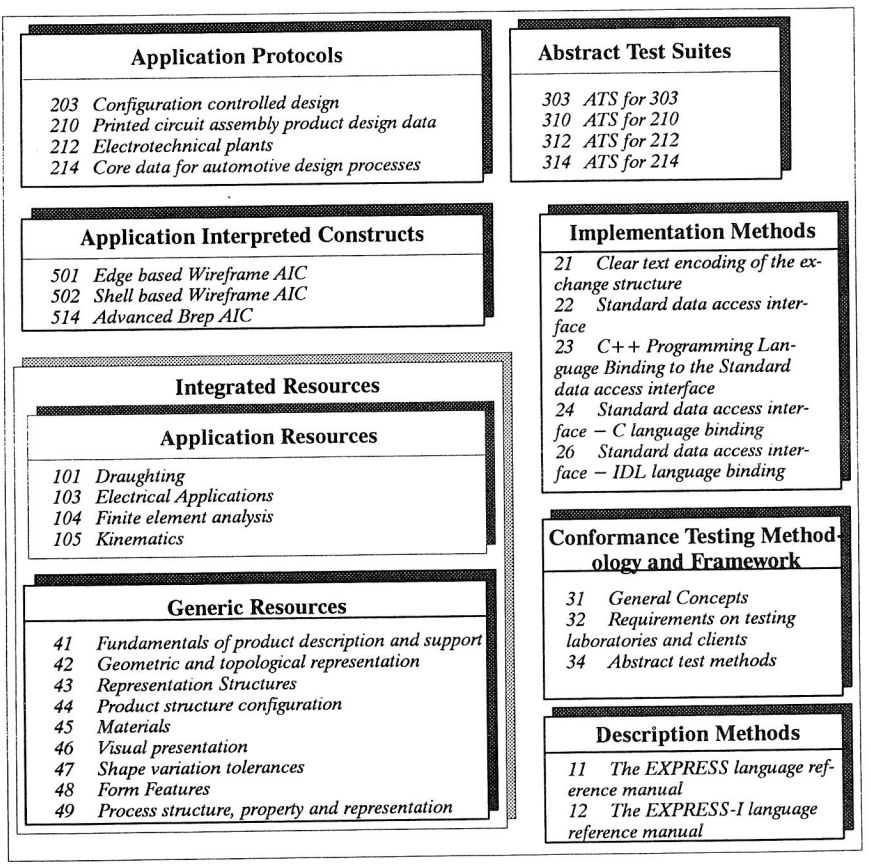


Abb. 3 Die Struktur von STEP – veranschaulicht durch die Strukturierung der Normdokumente

Innerhalb der Struktur von STEP kann im wesentlichen eine drei-Ebenen-Architektur erkannt werden, die in Anlehnung an die Entwicklung von Datenbanken nach ANSI/SPARC entstand /83/. Dabei entspricht der Ebene des externen Schemas in ANSI/SPARC /194/ den Anwendungsprotokollen (*application protocols*) in STEP. Das kon-

zeptionelle Schema in ANSI/SPARC kann mit den 'Integrierten Ressourcen' (*integrated resources*) identifiziert werden. Das interne Schema ist in Grenzen mit den Implementierungsmethoden (*implementation methods*) von STEP gleichzusetzen.

Im Folgenden sollen kurz die Partialmodelle die in Abb. 3 erkennbar sind, erläutert werden. Für die interessanteren Teile 'generic resources' und 'application protocols' soll dies etwas ausführlicher geschehen.

Description Methods (Beschreibungsmethoden)

Die in der Serie 10 der ISO-10303 Dokumentationen vorliegenden Teile der Norm beschreiben die Datenmodellierungssprache EXPRESS sowie eine Instantiierungsform von EXPRESS, EXPRESS-I. EXPRESS stellt für STEP eine derart grundlegende Basis bereit, daß diese Sprache, die für das weitere Verständnis grundlegend ist, in Kapitel 2.3 tiefgreifend erläutert werden soll.

Implementation Methods (Implementierungsmethoden)

In der Serie 20 sind Implementierungsmethoden beschrieben, die einerseits /43/ ein Austauschformat für die Übertragung von Daten zwischen verschiedenen Systemen mittels einer Datei sowie die Implementierung einer Zugriffsschnittstelle auf eine STEP-Datenbasis /44/ als auch Sprachanbindungen in C /46/, C++ /45/ und IDL/47/ auf diese Datenbasis definiert. Die Realisierung von Prozessoren für das Generieren und Lesen der Austauschdateien wird in Kapitel 3.3.2, die Realisierung von STEP-Datenbasen insbesondere auf Datenbanken ist in Kapitel 4.4.3, und die Realisierung von SDAI-Zugriffsschnittstellen in Kapitel 3.3.7.

Conformance Testing Methodology & Framework

Die Serie 30 besteht aus den Teilen /48/ und /49/, von denen das erste die Konzepte zur Konformitätsprüfung einer Implementierung der Norm ISO-10303 beschreibt. Darin sollen die Grundkonzepte einer standardisierten Prüfung von STEP-Implementierungen, insbesondere von Prozessoren hinsichtlich der Konformität dieser Implementierung gegenüber einem Anwendungsprotokoll festgelegt werden. Die Standardisierung der Testverfahren soll eine Vergleichbarkeit von Tests ermöglichen und die Akzeptanz der Testprotokolle, die von verschiedenen Institutionen erstellt, werden erhöhen. Die wiederholte Überprüfung eines Systems soll damit entfallen.

In /49/ werden die Anforderungen an Testinstitutionen und Implementierer dokumentiert, die für einen nachvollziehbaren Test nötig sind.

Integrated Resources: Application Resources (Anwendungsmodelle)

Die anwendungsspezifischen Basismodelle von STEP, dokumentiert in der Serie 100, sollen unter Nutzung der Basismodelle für spezielle Anwendungen gemeinsam nutzbare Elemente definieren. Die 'application resources' waren innerhalb der STEP Entwicklungsmethode als ein Gliederungsschema innerhalb einer modularen Synthese gedacht. Über die Problematik dieser Modularisierung mehr in Kapitel 6.3.

Abstract Test Suites (Abstrakte Testfälle)

In der Serie 300 (vormals Serie 1200) werden 'abstrakte' Testfälle definiert. Für jedes einzelne Anwendungsprotokoll wird ein zugeordneter Testfall festgelegt. Die Bezeichnung eines abstrakten Testfalls ergibt durch die Nummer des Anwendungsprotokolls, die für den zugeordneten Testfall um 300 erhöht wird.

Die Abstrakten Testfälle enthalten sowohl synthetische Testfälle sowie Tests, die industriellen Problemstellungen entnommen werden.

2.2.1 Generic Resources und Application Resources: Integrated Resources (Basismodelle)

Die Teile der Serie 40, die als sogenannte 'generic resources' bezeichnet werden, stellen einen wichtigen Grundstein innerhalb der ISO-10303 dar. In den 'generic resources' werden grundlegende Elemente des Produktdatenmodells beschrieben, die eine große Allgemeingültigkeit besitzen und zur Verwendung in anderen Teilen der Norm bereitgestellt werden.

Die einzelnen Teile der Serie 40 enthalten folgende Dokumente:

Teil 41, 'Fundamentals of Product Design and Support', beschreibt organisatorische Modelle zur Entstehung und Verwendung dieses Teils der Norm, Modelle zur Produktdefinition, den Produkteigenschaften und der Darstellung dieser Produkteigenschaften. Darüberhinaus werden hierin Verweise zu externen Spezifikationen wie Normen außerhalb der ISO-10303 ermöglicht. Darüberhinaus läßt sich der Änderungsdienst von Produktinformationen, die Zertifikation von Produkten, Freigabe von Konstruktionsinformationen, Vertragsinformationen, Sicherheitsinformationen, personenbezogene Informationen, zeitbezogene Informationen, globale, allgemeine und spezifische Maßeinheiten, Produktgruppeninformation und Hilfsinformationen beschreiben.

In Teil 42, 'Geometric and Topological Representation' werden die für die geometrische Produktbeschreibung grundlegenden Elemente beschrieben. Dabei wird zwischen Geometrie, Topologie und aus diesen Teilmodellen gebildeten Gestaltmodellen unterschieden. Jedes Basiselement ist dabei sowohl als zwei- oder dreidimensionales Element einsetzbar, was durch die Optionalität eines Datums der dritten Dimension erreicht wird. Alle Geometrieelemente sind dabei ausschließlich in einem rechtshändigen, kartesischen Koordinatensystem definiert.
Die Bedeutung des Teiles 42 für die Beschreibung von Gestaltmodellen ist bereits hier deutlich zu unterstreichen. Die Nutzung dieser Gestaltmodelle innerhalb von Datenverwaltungs- bzw. Austauschszenarien wird in den Kapiteln 5 sowie in Anwendungssystemen in Kapitel 7 vertieft.

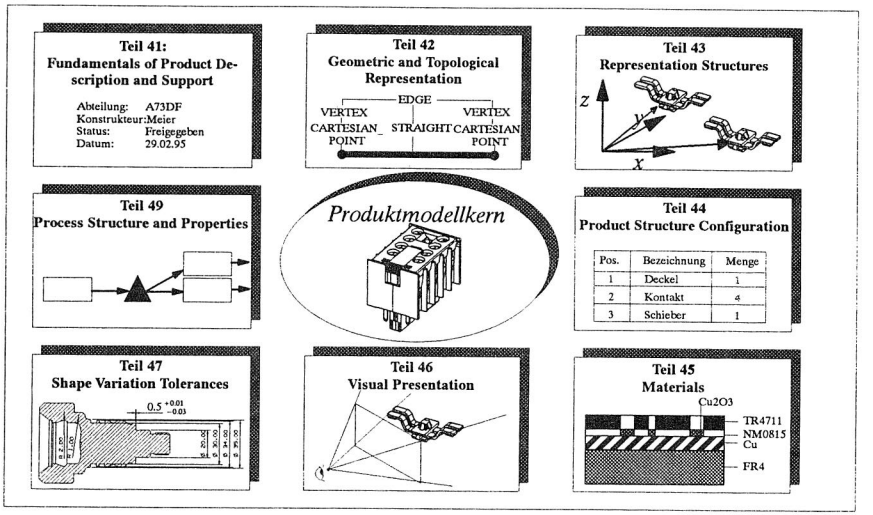


Abb. 4 Kern des STEP Produktmodells, gebildet durch die Teile der 'Generic Resources'

In Teil 43 'Representation Structures' wird die Darstellung von Elementen sowie deren räumlich Beziehung untereinander definiert. Damit wird die Zuordnung mehrerer Repräsentationen zu z.B. einem Koordinatensystem ermöglicht.

Innerhalb Teil 44 'Product Structure Configuration' werden Grundbegriffe der Produktstrukturierung, wie Stückliste, Teileverwendungsnachweis, Erzeugnisgliederung aber auch Arbeitspläne und Änderungsdefinitionen festgelegt. Obwohl hiermit versucht wird, eine einheitliche Nomenklatur und Bedeutung für diese Begriffe zu schaffen, wird die Produktstrukturierungsproblematik in den Anwendungsprotokollen nicht einheitlich gelöst. Ein allgemein akzeptiertes Basismodell wurde hiermit im Gegensatz zu den Geometrie-/Topologiemodellen nicht geschaffen.

Der Teil 45 'Materials' ermöglicht die Beschreibung von Materialeigenschaften sowie deren Darstellung für Produkte, Halbzeuge oder Grundstoffe. Die Materialbeschreibung ist auf eine deskriptive Modellierung der Eigenschaften des Materials bzw. der Basisstoffe eines Produkts orientiert.

In Teil 46 'Visual Presentation' wird die bildliche Darstellung des Produktmodells definiert. Dabei sind Kameramodelle und Beleuchtungsmodelle beschreibbar.

Teil 47 'Shape Variation Tolerances' ermöglicht die Beschreibung von Toleranzen. Dabei wird im wesentlichen die Toleranzdefinition der ISO286 übernommen.

Der Teil 48 'Form Features Information Model' sollte die Gestaltinformation über wiederkehrende Muster durch Formfeatures definieren. Die Entwicklung diese Basismodells

wurde jedoch Anfang 1995 eingestellt. Die Beschreibung von Features wurde daraufhin in die Gestaltbeschreibung der Anwendungsprotokolle selbst übernommen.

Der Teil 49 'Process structure and properties' soll die Beschreibung von Arbeitsplänen erlauben.

Obwohl mit der Definition der Serie 40 versucht wurde, ein modulares, aber dennoch integriertes Produktmodell zu schaffen, blieben einige Probleme hierbei ungelöst: Die Beschreibung von Features wurde zunächst innerhalb dieser Kernfunktionalität versucht, später aber mangels Akzeptanz aufgegeben. Grund hierfür mag die Problematik einer anwendungsfreien Definition von Features zu sein, die in krassem Widerspruch zum Verständnis von Features steht. Ähnliche Schwierigkeiten wurden bei der Definition eines Partialmodells für parametrische Modelle gefunden. Dies mag an dem prinzipiell unterschiedlichem Verständnis für einerseits evaluierte und nicht evaluierte Modelle liegen (siehe dazu auch Kapitel 5.1). Das Verständnis der Modellierungsmethodik ist darüberhinaus weithin unterschiedlich, wie in den analytisch-mathematischen Modellen der Geometrie-/Topologiemodellen sowie den deskriptiven Modellen der Materialmodellen zum Ausdruck kommt.

2.2.2 Application Interpreted Constructs (Anwendungsinterpretierte Konstrukte)

Über die Basismodelle hinaus sind in den AICs (Application Interpreted Constructs) höherwertige Schemata definiert worden. Diese stellen eine Gruppe von Elementen aus den Basismodellen dar, die unter Hinzunahme weiterer Randbedingungen Konzepte definieren, die in mehreren Anwendungsprotokollen benutzt werden sollen. Sie sind zunächst ohne eine weitere Anwendungsorientierung versehen. Hinsichtlich ihrer Bedeutung sind die AICs den anwendungsspezifischen Basismodellen sehr ähnlich. In der STEP-Entwicklungspraxis scheinen sie diese tatsächlich zu ersetzen.

Zunächst wurden die AICs nicht im Hinblick auf eine spätere Normierung entwickelt, sondern sollten zunächst als Zusätze der Norm beigefügt werden. Aus der Problematik der Wiederverwendung vorhandener Module heraus, werden diese Teilmodelle nun doch direkt als Serie 500 in die Norm integriert. Das damit erhoffte Lösungspotential sowie verbleibende Modellierungsschranken werden in Kapitel 6.3 verdeutlicht.

2.2.3 Anwendungsprotokolle

Anwendungsprotokolle spielen in STEP ein zentrale Rolle. Innerhalb eines Anwendungsprotokolls wird wird der Gültigkeitsbereich der Kontext und die Anforderungen einer Anwendung definiert /40/. Jede Implementierung eines Softwaresystems, das mit

Hilfe von Datenaustauschprozessoren mit anderen Systemen interagiert, oder seine Datenhaltung über ein SDAI abwickelt, muß sich immer an zumindest einem Anwendungsprotokoll orientieren. D.b., daß alle STEP-Implementierungen die Implementierung eines oder mehrerer Anwendungsprotokolle sein müssen.

Die Serie 200 definiert zum aktuellen Zeitpunkt 39 Anwendungsprotokolle, die jedoch in ihrem Inhalt nicht beschrieben werden sollen. In Kapitel 6 sollen einige Anwendungsprotokolle hinsichtlich ihres Gültigkeitsbereiches untersucht werden, um für das in Kapitel 7 prototypisch realisierte Anwendungssystem eine Modellbasis zu bieten.

Anwendungsprotokolle stellen somit einen Rahmen für die Implementierung der ISO-10303 bereit. Zusammen mit einer Implementierungsmethode kann für jedes Anwendungsprotokoll die Realisierung einer STEP-konformen Anwendung vorgenommen werden.

Für die Teile der Serie 200, den sogenannten Anwendungsprotokollen existiert eine vorgeschriebene Entwicklungsmethodik /121//172//173//174/. Insbesondere wird dabei die Erstellung der zu den Anwendungsprotokolle nötigen Inhalte beschrieben. Mit dem Verständnis über den Inhalt der Anwendungsprotokolle können dann die in Kapitel 6 vorgenommenen Erweiterungen eingeordnet werden.

Ein Anwendungsprotokoll wird in sechs Abschnitten beschrieben (Abb. 5):

- Durch den Anwendungsbereich (*scope*), der durch eine textuelle Beschreibung des Problembereiches der Anwendung erfolgt.
- Durch Verweisen auf andere Normen, die in der zu beschreibenden Norm benutzt oder referenziert werden.
- Mit Hilfe von Begriffsbestimmungen aus benutzten Normen sowie aus dem Anwendungsbereich.
- Aus den Anforderungen an das Anwendungsprotokoll, das dieses erfüllen muß, um die Tätigkeiten, die innerhalb des Anwendungsbereiches durchgeführt werden, informationstechnisch beschreiben zu können. Diese Anforderungsdefinition wird innerhalb des APs Anwendungsreferenzmodell (*Application Reference Model*, ARM) genannt. Die geforderten Informationseinheiten werden in sogenannten Funktionaleinheiten (*units of functionality*) gruppiert. Zur Beschreibung dieses ARM können formale Beschreibungsmethoden wie NIAM, IDEF1X, oder EXPRESS/EXPRESS-G benutzt werden, vorgeschrieben sind diese formalen Beschreibungen hier jedoch nicht.
- dem anwendungsinterpretierten Modell (*application interpreted model*, AIM), das mit Hilfe der Basismodelle die Anforderungen aus dem ARM "interpretiert". Diese "Interpretation" der Anforderungen aus dem ARM wird durch eine Identifizierung geforderter Informationselemente mit vordefinierten Informationselementen aus den Basismodellen vorgenommen. Die Identifizierung wird innerhalb der AP durch

eine sogenannte Abbildungstabelle (*mapping table*) dokumentiert, aus der hervorgeht, welches Informationselement der ARM mit welchem Informationselement des AIM korrespondiert.
Das AIM wird in EXPRESS beschrieben und ist innerhalb des Standards in langform enthalten. Dies bedeutet, daß alle Konstrukte, die aus den Basismodellen übernommen wurden, in das Schema des Anwendungsprotokolls kopiert werden und nicht, wie mit Hilfe der EXPRESS-Konstruktionen "REFERENCE FROM" oder "USE FROM" aus den Basismodellen referenziert werden. Der Grund hierfür ist eine erleichterte Benutzung der AIM-Schemas, da hierdurch Querverweise auf andere Dokumente hinfällig werden. Allerdings geht hierdurch eine Modularisierung der AIM-Schemas verloren, was unter anderem Auswirkungen auf verschiedene Implementierungstechniken hat (siehe Kapitel 6).

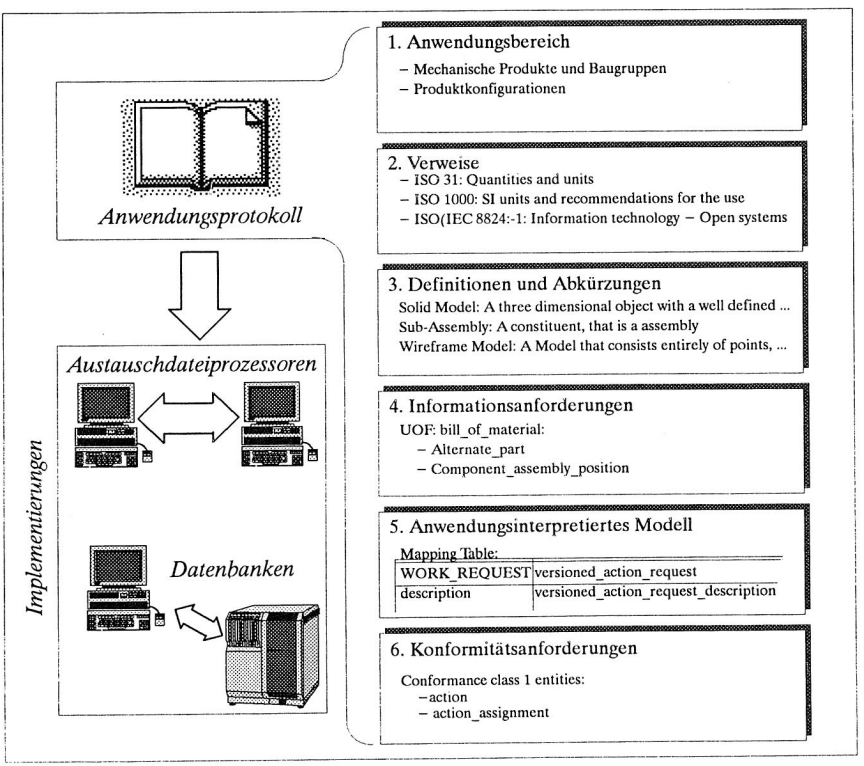


Abb. 5 Teile der Dokumentation eines STEP-Anwendungsprotokolls

- und schließlich den Anforderungstabellen für Konformität verschiedener Implementierungsstufen (*Conformance Classes*).

2.3 EXPRESS

Für das Verständnis der Norm ISO-10303 ist die Sprache EXPRESS von derart eminenter Bedeutung, daß ihrer Beschreibung in diesem Rahmen ein gewisser Umfang eingeräumt werden muß. Eine vollständige Beschreibung kann hier jedoch nicht erfolgen, jedoch sollen die für die Implementierung relevanten Eigenschaften der Sprache verdeutlicht werden.

EXPRESS ist eine formale Datenmodellierungssprache /41//189/, die zum Zweck der modellhaften, abstrakten und implementierungsunabhängigen Spezifikation von *statischen* Sachverhalten entwickelt wurde. Sie entlehnt Paradigmen und Konstrukte vielen bekannten Programmier- und Interaktionssprachen wie Ada, Algol, C, C++, Euler, Modula-2, Pascal PL/I und SQL. Sie stellt jedoch in ihrer Gesamtheit eine völlig neuartige Sprache dar, die in einigen Bereichen Konzepte verwirklicht, die sonst in keiner Sprache zu finden sind. EXPRESS wurde mit dem Ziel entwickelt, eine gleichermaßen maschinell verarbeitbare und vom Menschen lesbare Sprache zu sein. Sie stellt im Gegensatz zu Programmiersprachen jedoch keine Laufzeitsemantik zur Verfügung. EXPRESS ist damit eine Sprache zur Spezifikation von statischen Sachverhalten.

2.3.1 Aufbau von EXPRESS

EXPRESS stellt sieben deklarative Konstrukte bereit, die im folgenden kurz dargestellt werden:

Schemas

Ein Schema in der Begriffswelt von EXPRESS stellt einen abgegrenzten Interessenbereich (universe of discourse) bereit. Innerhalb eines solchen Interessenbereiches werden Konstrukte niederer Hierarchie definiert, die einen größeren Sinnzusammenhang besitzen. Es ist möglich, über sogenannte Schemareferenzen (REFERENCE FROM, USE FROM), Konstrukte aus anderen Schemas in das aktuelle Schema zu importieren.

Typen:

In EXPRESS ist es möglich neue Typen durch Definition einzuführen. Die Definition kann sich auf Basistypen oder bereits definierte Typen beziehen. In EXPRESS vordefiniert sind die Typen REAL für Gleitkommazahlen, INTEGER für ganze Zahlen sowie der Typ NUMBER für einen allgemeinen Typ von Zahlen. Darüberhinaus stehen die Typen STRING für Zeichenketten sowie BINARY für Ketten von 0/1-Werten bereit. EXPRESS kennt zwei Arten logischer Typen: Der zweiwertige Typ BOOLEAN erlaubt die Werte FALSE und TRUE, der dreiwertigen Typ LOGICAL die Werte FALSE, UNKNOWN und TRUE. Als aggregierte Typen können die Typen ARRAY als Vektortyp, SET als Mengentyp, LIST als Liste und BAG als Mengentyp, der Elemente mehrfach enthalten kann, verwendet werden. Des weiteren kennt EXPRESS den Aufzähltyp ENUMERATION, der aus

einer Auswahl von Bezeichnern besteht, sowie den Typ SELECT, der einen Typ definiert, der die Vereinigungsmenge der Elemente des Typs darstellt.

Entitäten:

Entitäten (ENTITY) stellen innerhalb EXPRESS das wesentliche Ausdrucksmittel bereit. Mit ihrer Hilfe ist es möglich Datenelemente sowie Beziehungen zwischen einzelnen Datenelementen zu beschreiben. Mit:

```
ENTITY a;
END_ENTITY;
```

wird eine Entität mit Namen *a* vereinbart. Dabei kennt EXPRESS die Möglichkeit hierarchische Beziehungen zwischen Entitäten zu definieren, wobei Paradigmen objektorientierter Modellierung wie Vererbung und Spezialisierung zur Anwendung kommen. Das Vererbungsprinzip von EXPRESS ist äußerst ausdrucksstark. So ist mehrfache Vererbung möglich, wobei die Sprachdefinition Regeln für Vererbungskonflikte definiert, wie sie durch mehrfache Vererbung von Attributen gleichen Namens hervorgerufen werden können.

```
ENTITY a;
END_ENTITY;

ENTITY b
SUBTYPE OF a;
END_ENTITY;
```

definiert zwei Entitäten *a* und *b*. Die Entität *b* stellt dabei einen Subtyp (Unterklasse, Spezialisierung) der Entität *a* dar. Eine Besonderheit des Vererbungsprinzips von EXPRESS ist die implizite Definition komplexer Objekte durch die Angabe einer Auswahl von Supertypen (Entitytypen) durch die Verkettungsoperatoren AND (und), ANDOR (inklusives oder) und ONEOF (exklusives oder)/13/. Durch die kombinatorische Verknüpfung der in der SUPERTYPE-Klausel aufgezählten Entitytypen werden Entitäten definiert, die innerhalb der Modellwelt existieren, dort jedoch nicht explizit benannt werden.

```
ENTITY a;
SUPERTYPE OF (b ANDOR c)
END_ENTITY;

ENTITY b
SUBTYPE OF a;
END_ENTITY;

ENTITY c
SUBTYPE OF a;
END_ENTITY;
```

definiert somit explizite Entitäten mit Namen a, b und c sowie implizit Entitäten, die als komplexe Entitäten 'a&b', 'a&c' und 'a&b&c' bezeichnet werden.

Eine Entität wird durch ihren Bezeichner sowie die innerhalb der Entität definierten Attribute bestimmt. Ein Attribut selbst kann innerhalb einer Entität als optional (OPTIONAL) gekennzeichnet sein, was bedeutet, daß ein Wert innerhalb einer Instantiierung nicht notwendigerweise zur Verfügung gestellt werden muß. Werte von Attribute können des weiteren als abgeleitete Größen (DERIVE) bestimmt werden. Die Beziehung zwischen Datenelementen wird innerhalb einer Entität durch die Definition inverser Attribute (INVERSE) ermöglicht. Dadurch wird die Verwendung einer Entität nur im Zusammenhang mit einer anderen ermöglicht. Mit Hilfe der UNIQUEness-Klausel kann eine Menge von Attributen (bzw. deren Werte) für Instanzen einer Entität eindeutig gemacht werden /41/:

```
ENTITY person
ABSTRACT SUPERTYPE OF (ONEOF(female, male));
first_name : STRING;
size : OPTIONAL REAL;
birth_date : date;
children : SET [0:?] OF person;
DERIVE
age : INTEGER := years(birth_date);
END_ENTITY;

ENTITY male
SUBTYPE OF (person);
wife : OPTIONAL female;
END_ENTITY;

ENTITY female
SUBTYPE OF (person);
INVERSE
husband : SET [0:1] OF male FOR wife;
END_ENTITY;
```

Die oben definierte Entität 'person' besitzt somit ein Attribut mit Namen 'name' vom Typ STRING, ein optionales Attribut mit Namen 'nickname' vom Typ STRING, ein abgeleitetes Attribut mit Namen 'age' vom Typ REAL, dessen Berechnungsvorschrift aus einer hier nicht weiter interessierenden Funktion 'years' besteht[3].
Die von 'person' abgeleitete Entität 'male' besitzt ein optionales Attribut 'wife', das selbst vom Typ 'female' ist. Hiermit wird eine Beziehung zwischen den Entitäten 'male' und 'female' formuliert. Diese Beziehung wird in der Entität 'female' durch ein inverses

3. *Die Diskussion um die Semantik von EXPRESS führt u. a. zu der Frage, ob die in EXPRESS formulierbaren Ausdrücke abgeleiteter Attribute, zur Bestimmung von Werten oder zur Definition einer Konsistenzbedingung dienen sollen. Im vorliegenden Fall würde damit der Wert des Attributes 'age' nicht aus dem gegebenen Ausdruck berechnet werden, sondern ein gegebener Wert von 'age' durch den Ausdruck hinsichtlich seiner Korrektheit validiert werden und damit die Korrektheit einer Instanz vom Typ a überprüft werden. Obwohl die zweite Sicht wesentlich stärker dem Ansatz von EXPRESS als einer Modellierungssprache entspricht, lassen die Überführungsmethoden von EXPRESS-Modellen auf STEP-Übertragungsdateien sowie das SDAI-Binding die erstgenannte Semantik als gegeben erscheinen. Ein Beweis für diese Annahme ist dies jedoch nicht.*

Attribut 'husband' ausgedrückt, das angibt, daß keine oder eine Instanz[4] vom Typ 'male' vorhanden sein kann, und die Beziehung über das Attribut 'wife' der Entität 'male' erreicht wird.
Eine Entität kann somit im weitesten Sinn als ein *Entitytyp* verstanden werden.

Konstanten:

Eine Konstante (CONSTANT) in EXPRESS stellt innerhalb des Sprachumfanges den konstanten Wert eines benannten Typs bereit. Eine EXPRESS Konstante kann von jedem in EXPRESS definierbaren Typ sein.

Funktionen und Prozeduren

Funktionen (FUNCTION) und Prozeduren (PROCEDURE) stellen algorithmische Sprachanteile bereit, wobei Funktionen einen Wert beliebigen Typs zurückliefern. Die Übergabeparameter können für Prozeduren als veränderbar (VAR) gekennzeichnet werden. Seiteneffekte durch Funktionen oder Prozeduren sind per definitionem ausgeschlossen. EXPRESS definiert in seinem Sprachumfang eine Vielzahl von internen Funktionen auf unterschiedliche Entitytypen (z.B. SIZEOF als Funktion zur Bestimmung der Anzahl der Elemente einer aggregatwertigen Instanz) sowie zwei Prozeduren (REMOVE und INSERT), die auf aggregatwertigen Typen definiert sind.

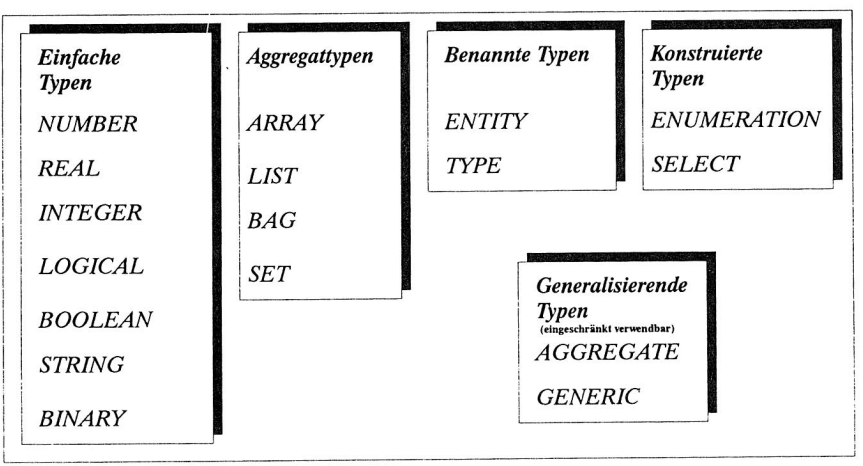


Abb. 6 Klassifikation der internen EXPRESS-Datentypen

Regeln:

Globale Regeln (RULE) stellen eine Möglichkeit dar, die Modellwelt innerhalb eines Schemas durch Constraints einzuschränken. Regeln können auf Entitäten innerhalb ei-

4. *Die hier gezeigte Modellierung der Kardinalität der Beziehung durch eine Menge von 'male' wird wegen der nicht modellierbaren Optionalität der inversen Beziehung notwendig. Damit wird jedoch eine inhaltlich nicht zutreffende Modellbildung über die Menge von 'male' vorgenommen, die sachlich nicht gerechtfertigt erscheint.*

nes Schemas definiert werden und liefern, abhängig von einer Berechnung, einen logischen Wert zurück, der bestimmt, ob Instanzmengen der Entität eines Schema bezüglich dieser Regel gültig sind oder nicht.

EXPRESS kennt sowohl für Entitytypen als auch für definierte Typen sogenannte lokale Regeln (WHERE), die sich entweder auf ein Attribut der Entität oder auf den Wert einer Variable des Typs beziehen. Diese lokale Regel liefert einen logischen Wert, der eine zusätzliche Einschränkung für den zugrundeliegenden Typ definiert.

Die in Abb. 6 dargestellte Einteilung der EXPRESS Datentypen hält sich an die in /41/ vorgenommene Gliederung. Diese Gliederung entspricht jedoch keinesfalls einem typisierenden Klassifikationsschema. Der Datentyp NUMBER z.B. kann als ein Vereinigungstyp der Typen INTEGER und REAL angesehen werden. (Diese Klassifikation liegt z.B. der Abbildung der C++-Sprachanbindung des SDAI /45/ zugrunde.) Des weiteren kann der Datentyp INTEGER als eine Spezialisierung des Datentyps REAL angesehen werden, wie dies auch in den EXPRESS-Metamodellen /196/ vorgenommen wird.

In EXPRESS sind sieben Klassen von Operatoren definiert, die in

- arithmetische ("+", "−", "*", "/", "**", "DIV", "MOD"),
- relationale ("=", "<>", ">", ">=", "<=", ":=:", ":<>:", "IN", "LIKE"),
- String-Operatoren ("[]", "+"),
- logische Operatoren ("NOT", "AND", "OR", "ANDOR"),
- Aggregat-Operatoren ("[]", "*", "+", "−", "<=", ">=", "QUERY", "=", "<>", "IN")
- Komponenten-Operatoren (".", "\", "=", "<>", ":=:", ":<>:", "||")

unterschieden werden können. Sie können weiterhin in

- einwertige('NOT', unäres '−' und unäres '+'),
- zweiwertige (z.B. binäres "+") und
- dreiwertige ('[:]', Aggregatausschnitt)

Operatoren klassifiziert werden.

Um den Ablauf in den algorithmischen Anteilen der Sprache zu steuern, bietet EXPRESS die folgenden Anweisungen an: ALIAS, Zuweisung (":="), CASE, Verbundanweisung (BEGIN, END), ESCAPE, IF – THEN – ELSE, NULL, Prozedur- und Funktionsaufruf, REPEAT, RETURN und SKIP, die jeweils analogen Anweisungen höherer Programmiersprachen entsprechen.

Obwohl EXPRESS mit Funktionen, Prozeduren und Regeln prinzipiell mächtige algorithmische Sprachanteile bereitstellt, ist durch EXPRESS kein Anweisungsablauf im Sinne eines Programmes einer konventioneller Programmiersprachen definiert. Die al-

gorithmischen Sprachanteile sind nur zur weiteren Charakterisierung eines Informationsmodells bereitgestellt. Im Gegensatz hierzu wird versucht, in einer Vielzahl von EXPRESS-Dialekten dieses von vielen Seiten identifizierte Defizit mit Erweiterungen der EXPRESS-Sprachsyntax aufzulösen (siehe Kapitel 2.3.3). Die damit verbundene Verschiebung der Anwendungszentrierung von EXPRESS wird damit jedoch weitgehend ignoriert.

Bei der Definition von EXPRESS war eine rechnerorientierte Verarbeitbarkeit vorgesehen. Zur Definition der Sprache EXPRESS liegt daher in /41/ eine lexikalische Sprachdefinition in WSN (Wirth Syntax Notation) vor, die im Hinblick auf die Realisierung von Parsern abzielt.

2.3.2 Das graphische EXPRESS-Äquivalent EXPRESS-G

Neben der lexikalischen Definition der Sprache EXPRESS existiert eine graphische Notation als Untermenge von EXPRESS, die im Anhang von /41/ als EXPRESS-G definiert wurde. EXPRESS-G verzichtet auf die Definition der algorithmischen Anteile und präsentiert die Möglichkeit, die Struktur der Datenelemente hinsichtlich ihrer hierarchischen Struktur sowie ihrer Referenzierung grafisch zu repräsentieren.

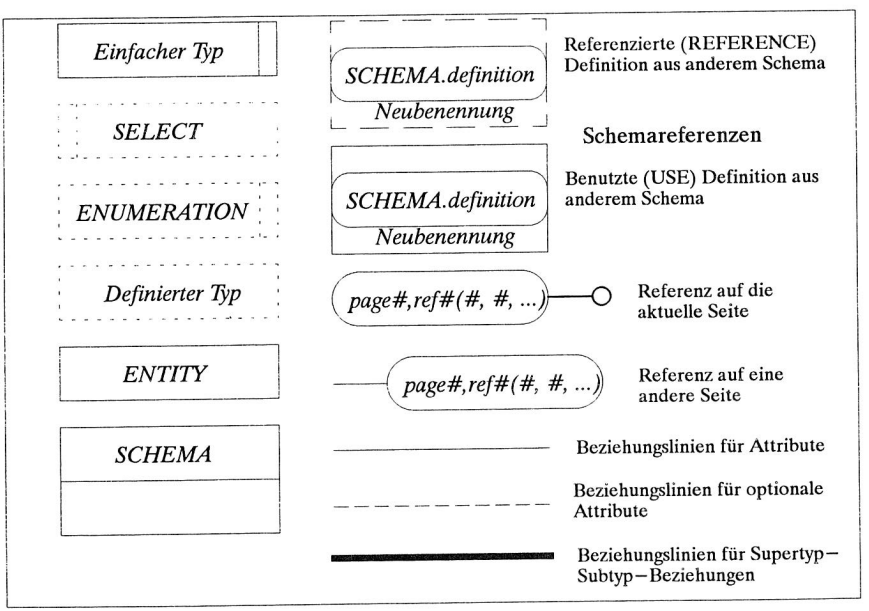


Abb. 7 Elemente der graphischen Notation von EXPRESS – EXPRESS-G

Die in Abb. 7 dargestellten Elemente stellen die für EXPRESS-G benutzbaren Elemente dar. Für die linksseitig angegebenen typbezeichnenden Elemente sollen die aktuellen Typbezeichner eingesetzt werden.
Die im Modellschema vorhandenen Beziehungen werden durch Linien zwischen den Elementen repräsentiert. Die Beziehung einer Entität zu einem optionalen Attribut wird mit einer gestrichelten Linie dargestellt, ebenso wie Schema-Schema Referenzen. Supertyp-Subtyp Beziehungen finden Repräsentationen in fetten Linien, alle anderen Beziehungen in durchgezogenen Linien. Beziehungen sind prinzipiell bidirektional, jedoch ist eine der Beziehungsrichtungen betont. Die betonte Beziehungsrichtung für ein explizites Attribut einer Entität ist die Richtung auf das Attribut, für die Supertyp-Subtyp Beziehung zum Subtyp hin. Die Betonung der Beziehungsrichtung wird durch einen Kreis in der Richtung der Betonung, am Ende der Beziehungslinie, verdeutlicht.

Die Rolle eines Attributs innerhalb einer Entität, repräsentiert durch den Attributnamen soll innerhalb eines EXPRESS-G Diagramms den Beziehungslinien zwischen Entität und Attribut zugeordnet werden.
Über den Sprachumfang von EXPRESS hinaus erlaubt EXPRESS-G die Verbindung von Teildiagrammen über Seiten hinweg mit Hilfe sogenannter Seitenverweise (page references).

Obwohl EXPRESS-G Diagramme eine Verdeutlichung eines EXPRESS Modells ermöglichen soll, sind die Ausdrucksmöglichkeiten innerhalb der Notation sehr begrenzt. Zum einen gibt es keine vorgeschriebene Vorzugsrichtung für bestimmte, innerhalb des Modells dargestellte Sachverhalte. Z.B. werden Supertyp-Subtyp Beziehungen nicht notwendigerweise in einer streng horizontalen oder vertikalen Richtung angeordnet. Die Anordnung der Beziehungen an ein Element ist darüberhinaus ebenfalls frei wählbar und daher bestimmte Beziehungstypen schwer visuell erfaßbar.
Komplexe Diagramme leiden darüberhinaus an der Problematik kreuzender, aber logisch beziehungsloser Verbindungslinien, deren logische Verbindungsfreiheit nicht dargestellt werden kann.

2.3.3 Erweiterungen und Dialekte von EXPRESS

Aus verschiedenen Gründen sind über die Definitionen in /41/ hinaus unterschiedliche Dialekte von EXPRESS entstanden. Von diesen ist bisher nur **EXPRESS-I** in den Status eines Normvorschlages übernommen worden. EXPRESS-I erlaubt die Beschreibung von Instantiierungen von Entitäten, die in EXPRESS definiert wurden. Ziel ist im wesentlichen die Vereinfachung von Tests von EXPRESS-Schemata mit Hilfe einer EXPRESS-ähnlichen Syntax.

In /108/ wird die Definition einer Schema-Mapping Sprache **EXPRESS-M** vorgestellt, die es ermöglicht, einen algorithmisierten Übergang zwischen verschiedenen Versio-

nen eines Schemas mittels einer an EXPRESS angelehnten Syntax zu beschreiben. Wesentliche Idee ist es hierbei, eine Implementierung eines Schemas vorzunehmen, und eventuell später auftretende Änderungen an diesem Schema nicht innerhalb der Implementierung zu ändern, sondern mit Hilfe einer Schema-Änderungsbeschreibung in EXPRESS-M die Änderung außerhalb der Implementierung abzufangen. Die Änderungen innerhalb eines STEP-Schemas wird jedoch bisher nur in der Entwicklungsphase eines Anwendungsprotokolls aus Gründen der Fehlerhaftigkeit oder nicht erfüllter Anforderungen der STEP-Modelle vorgenommen. Sollte eine Implementierung jedoch diese schemainhärenten Fehler implementiert haben, so ist bei Anpassung des EXPRESS Schemas auch eine Bereinigung der Implementierung angeraten. Darüberhinaus ist die Fähigkeit zur Abbildung der in EXPRESS nutzbaren Konstrukte begrenzt.

In /112/ wird ein EXPRESS-M ähnlicher Dialekt als **EXPRESS-V** (Views) entwickelt, der ähnlich den Sichten in relationalen Datenbanken unter Definition von Mappingvorschriften neue Sichten auf Datendefinitionen in EXPRESS-Datenbasen erlaubt. In Kapitel 7 wird ein für die Entwicklungsmethodik der STEP Anwendungsprotokolle wichtiger Anwendungsfall von EXPRESS-V diskutiert.

Da EXPRESS im wesentlichen auf die Definition statischer Sachverhalte ausgelegt ist, wurde früh das Defizit zur Beschreibung dynamischer Vorgänge erkannt. Aus diesem Grund wurden Erweiterungen von EXPRESS zur Definition von Prozeßdefinition in /90/ und /110/ in **EXPRESS-P** vorgenommen. Dabei werden im Wesentlichen Möglichkeiten der Kommunikation zwischen Kommunikationspartnern mit Hilfe von Signalen über Kommunikationspfade definiert. EXPRESS-P ist dabei eine direkte Übermenge von EXPRESS.

Die EXPRESS Dialekte **EXPRESS-C** und **EXPRESS-R** wurden innerhalb des ESPRIT-Projektes PISA /179/ entwickelt und entsprechende Werkzeuge (Parser, Compiler, graphische Modellierungswerkzeuge usw.) hierfür realisiert.
EXPRESS-R (**r**equirement specification) erweitert EXPRESS um das Konstrukt der Relation, um Verhaltensmodellierung von Entitäten mit Hilfe von Operationen mit Vor- und Nachbedingungen, um Ereignissen, die Transaktionen auslösen sowie um Möglichkeiten unvollständig definierte Typen zu vereinbaren.
EXPRESS-C (**c**onceptual design) erweitert EXPRESS um die Verhaltensmodellierung von Entitäten (implizit definierte, benutzerdefinierte und globale Operationen, Vor- und Nachbedingungen, Ausnahmen und dynamische Typänderung), die Spezifikation eines dynamischen Modells (Ereignisse, Trigger und Aktionen) sowie um ein Ausführungsmodell (Aktivitätsmechanismen, Integritätsüberprüfung und Ausnahmebehandlung). Durch die induzierten Erweiterungen von EXPRESS-R und EXPRESS-C, die in Erweiterungsvorschläge für EXPRESS-2 eingegangen sind, wird die Ausrichtung von EXPRESS als Modellierungssprache aufgegeben und damit in die Nähe einer Programmiersprache gesetzt. Neben dem Defizit einer Abbildung der Sprache EXPRESS bzw. EXPRESS-C auf ein Maschinenmodell erscheint vor allem die damit ermöglichte Festlegung eines Verhaltens von Objekten und damit der Normierung des Verhaltens von Anwendungssystemen äußerst problematisch. Die Gefahr der Festlegung und damit der

negativen Stabilisierung von nicht oder wenig normierungsbedürftigen Systemeigenschaften wird hierdurch heraufbeschworen. Der Grundsatz der Normierung "So wenig wie möglich, so viel wie nötig" wird durchbrochen. Weiterhin werden durch die in PISA entwickelten Erweiterungen der Basisentwicklungen zur Definition von Anwendungssystemen genutzt. Jedoch wird nicht deutlich, wie die in der STEP Entwicklung vorgenommene Teilung in Anwendungsreferenzmodell und anwendungsinterpretiertes Modell durch die EXPRESS-Erweiterungen genutzt werden sollen. Tatsächlich wird in der PISA-Architektur aus den EXPRESS-C-Modellen direkt ein Anwendungsprogramm erzeugt. Eine Abbildung auf genormte Basismodelle wird nicht unterstützt.

2.3.4 Bewertung von EXPRESS als Modellierungssprache

Mit dem Ziel ein formale, rechnergestützt verarbeitbare Datenmodellierungssprache zu schaffen, wurde mit EXPRESS die erste von der ISO genormte Datenmodellierungssprache geschaffen. Die rechnergestützte Verarbeitbarkeit wurde durch die Spezifikation der Sprachsyntax in einer WSN (Wirth Syntax Notation) unterstützt, um die Generierung von Parsermodulen zu erleichtern.

Die formale Spezifikation zielt zwar im wesentlichen auf eine rechnergestützte Verarbeitbarkeit ab, jedoch sind auch hier einige Einschränkungen gegeben. Durch die Möglichkeit ein Symbol vor seiner Deklaration verwenden zu können werden für Parser einige unnötige architekturale Spezifika erwartet, die bei syntaktisch alternativer Sprachspezifikation vermeidbar gewesen wären (siehe Kapitel 3.3.1).

Mit EXPRESS können statische Informationsmodelle beschrieben werden. Instanzen dieser Informationsmodelle können die Regeln des für sie definierten Schemas erfüllen oder verletzen. Falls sie die Regeln verletzen, gelten sie als ein nicht gültiger Datensatz im Sinne des Schemas. Da für ein EXPRESS Schema nicht angegeben wird wann und wie die Überprüfung der modellierten Regeln vorzunehmen ist, bzw. welche Reaktionen innerhalb einer Implementierung zu treffen sind, falls diese Regeln verletzt werden, fehlt ein wesentlicher Teil einer Implementierungssemantik. Dieses Problem ist in Kapitel 2.4.2 weiter ausgeführt.

Die Orientierung von EXPRESS als Definitionssprache in STEP wird im wesentlichen durch die Definition von sogenannten 'Mapping Rules' für die Implementierungsmethoden bestimmt (Abb. 8)/40/. Diese architekturale Methodik in STEP stellt die wesentliche Neuerung für Implementierungsrichtlinien in der EDV-Welt dar. Diese formalen Mapping-Regeln sind für die bisher definierten Implementierungsmethoden in den Kapiteln 2.4.1 und 2.4.2 beschrieben.

EXPRESS wird oftmals /190/ als objektorientierte Sprache bezeichnet. Da EXPRESS jedoch keine Programmiersprache sondern eine Datenmodellierungssprache darstellt, können elementare Paradigmen objektorientierter Systeme, die auf ein Laufzeitverhal-

ten wie Ausnahmebehandlung, Objektkommunikation und spätes Binden hin orientiert sind, prinzipiell nicht unterstützt werden.

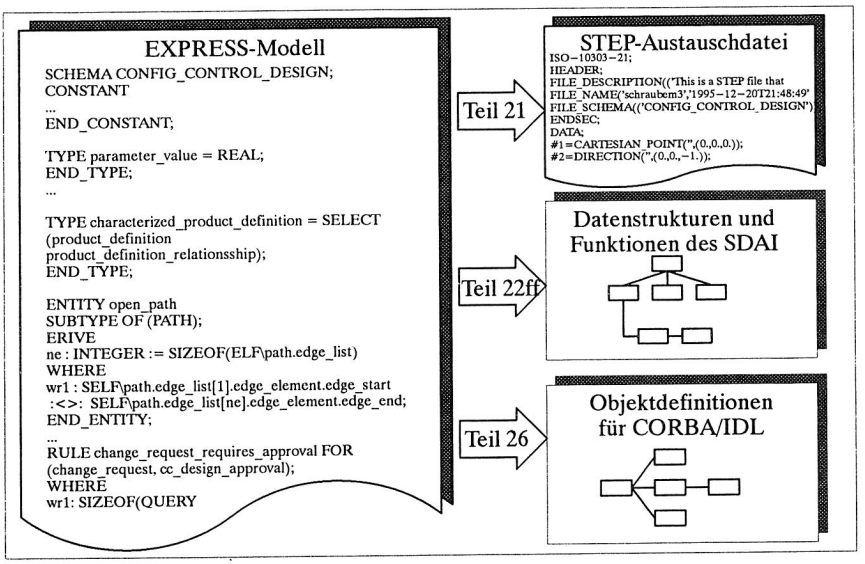


Abb. 8 *Die STEP Implementierungsmethoden definieren Mappingregeln von EXPRESS-Schemata auf Implementierungskontexte*

Weiterhin problematisch an EXPRESS ist der Verweis auf das Verhalten zu einer Laufzeit, wie die Übergabe eines undefinierten Wertes ("?") beim Auftreten von Fehlern in den algorithmischen Sprachanteilen. Da jedoch eine Laufzeit als solches für EXPRESS nicht definiert ist, kann dieses Verhalten auch nicht gefordert werden[5].

Auffälligste Schwäche von EXPRESS ist jedoch die Vermischung der Konzepte der Modellwelt mit der Implementierung. Während ein in EXPRESS definiertes Schema eine reine Modellwelt darstellt, können mit einigen Operatoren und den algorithmischen Anteilen der Sprache Instanzen der Modellwelt verarbeitet werden. Wie dies zu geschehen hat ist nicht der EXPRESS-Spezifikation zu entnehmen, sondern in den Implementierungsmethoden festgelegt.

5. *Daß Implementierungen wie EXPRESS-Parser bzw. Interpreter oder darauf aufbauende STEP-Austauschdateiparser einen derartiges Verhalten implementieren, kann hierfür natürlich nicht als Beweis der Definition dieser Laufzeit herangezogen werden.*

2.4 Implementierungsmethoden von STEP

Wie in Kapitel 2.2.3 erläutert, werden durch die Anwendungsprotokolle zusammen mit den Implementierungsmethoden und den Konformitätstests die Randbedingungen für eine nach STEP kompatible Implementierung eines EDV-Systems vorgegeben.

Neben den in /40/ vorgeschlagenen und in /43/ und /44/ mit /45/, /46/ und /47/ standardisierten Implementierungen werden in /40/ auch Implementierungen in Hauptspeicherstrukturen und Wissensbasen vorgeschlagen. Für diese Implementierungsformen besteht zur Zeit kein Standardisierungsvorhaben. In Kapitel 3.3 werden jedoch Ansätze zur Realisierung solcher Implementierungsformen gezeigt.

2.4.1 STEP-Austauschdateien

Die Entstehung von STEP gründet sich zunächst auf dem Wunsch nach einem neutralen Format zum **Austausch** von produktbeschreibenden Daten. Der Austausch solcher produktbeschreibenden Daten ist im wesentlichen an den Austausch von Informationseinheiten innerhalb von Dateien geknüpft. Obwohl innerhalb von STEP keine Orientierung auf Hardware- oder Systemsoftware genommen werden sollte, hat sich die Verwendung von Dateien als primäre Informationsstrukturierungseinheit durchgesetzt. Dies mag wohl in der weiten Verbreitung dateiorientierter Betriebssysteme im technischen Bereich wie UNIX bzw. seine Derivate, VMS oder MS-DOS sowie dessen Derivate liegen.

Die für den Austausch relevanten Informationseinheiten in einer STEP-Austauschdatei (physical file) sind die innerhalb Entitäten gruppierten Attribute bzw. deren Werte. Die Instanzen von EXPRESS-Entitäten sind damit die wesentlichen adressierbaren Elemente in einer Übertragungsdatei. Für eine EXPRESS Entität deklariert als:

```
ENTITY point;
x       :       REAL;
y       :       REAL;
z       :       OPTIONAL REAL;
END_ENTITY;
```

ist eine hierfür zu übertragende Instanz in einer Austauschdatei mit:

#1 = POINT (3.0, 5.0, 8.0);

zu schreiben. Der Instanzenidentifikator der durch einen Festkommawert größer 0 hinter dem Zeichen "#" als 1 angegeben wurde, ist der Bezeichner dieser Instanz, der jedoch nur innerhalb des übertragenen Datensatzes gültig ist, und darüberhinaus keine persistente Identifikation besitzt. Die Werte der innerhalb der Instanz übertragenen Attribute (für abgeleitete (DERIVE) und inverse (INVERSE) Attribute wird keine Übertragung

vorgenommen) werden in der Reihenfolge ihrer Definition innerhalb der Entität in der Instanz aufgelistet. Alle in einer Austauschdatei benutzten Typnamen sind unabhängig von der Schreibweise in dem EXPRESS-Modell mit Großbuchstaben zu schreiben.

Das Fehlen eines optionalen Attributs (OPTIONAL) wird durch das Sonderzeichen "$" markiert, redefinierte Attribute durch "*".

Die Ausprägung des Typs eines Attributes kann dann nicht notwendigerweise eindeutig aus dem Modell extrahiert werden, falls der Typ des Attributs einen SELECT Typ beinhaltet. In diesen Fällen muß der Typ des zu übertragenen Attributs unter Angabe des gewählten Selektionselementes typisiert werden.

```
ENTITY a;
attr1 : value;
attr2 : value;
END_ENTITY;

TYPE value = SELECT (rational, irrational); END_TYPE;

TYPE rational = REAL(3); END_TYPE;
TYPE irrational = REAL (5); END_TYPE;

#1=a(RATIONAL(3.14), IRRATIONAL(2.41));
```

In obigem Beispiel sind die Ausprägungen der Attribute der Instanz 1 von a vom Typ 'rational' für attr1 sowie 'irrational' für attr2. Die hierbei nötige Typisierung der beiden Gleitkommawerte 3,14 und 2.41 auf die jeweiligen Typen verhindert die durch den SELECT Typen 'value' mögliche Mehrdeutigkeit.

Für Instanzen, die von Entitäten gebildet werden, die einen Supertyp besitzen, sind zwei unterschiedliche Abbildungsverfahren nötig. Die Abbildung von Werten ererbter Attribute läßt sich prinzipiell unterschiedlich handhaben. Mit Hilfe des 'externen Mappings' werden ererbte Attribute in die Liste der zu übertragenden Attribute der Instanz eingereiht. Die Reihenfolge der Einreihung bei einer Mehrzahl von Supertypen wird durch die Sequenz innerhalb der Supertypdefinition vorgegeben:

```
ENTITY a;
attr1 : INTEGER;
END_ENTITY;

ENTITY b
SUBTYPE OF (a);
attr2 : REAL;
END_ENTITY;

#1=B(2,3.14);
```

Bei Anwendung des 'internen Mappings' wird eine Menge von Instanzen erzeugt, die jeweils die lokal definierten Attribute tragen. Mit obigem Beispiel gilt dann:

```
#1=(A(2),B(3.14));
```

Für komplexe Instanzen (Instanzen von komplexen Entitäten, siehe Kapitel 2.3.1) muß immer letzteres Verfahren angewandt werden, für alle alle anderen ist die Anwendung des Mappingverfahrens in Abhängigkeit der implementierten Konformitätsklasse wählbar[6].

Obwohl in der EXPRESS Modellwelt für Gleitkommazahlen eine optionale Angabe der Präzision dieser Gleitkommazahl möglich ist, wird für die Schreibweise von Gleitkommazahlen in der Übertragungsdatei abschließende Nullen ignoriert. Für jede Gleitkommazahl wird angenommen, daß diese nach den übertragenen Ziffern nur noch abschließende Nullen enthält, solange nicht die in der Präzisionsangabe angegeben Anzahl von Ziffern erreicht ist.

Innerhalb der Spezifikation der Übertragungsdatei sind die folgenden zwei Anomalien festzustellen:
Für die Austauschdatei wird ein Scope-Konzept eingeführt, das die Gruppierung von Instanzen erlaubt, und für diese Instanzenmenge einen gemeinsamen, abgeschlossenen Namensraum vergibt. Damit wird auf der der Ebene der Datensätze ein Modellierungkonzept zur Verfügung gestellt, das nicht durch äquivalente Modellbildung im Datenmodell gerechtfertigt ist, und eine beliebige Interpretation eines Anwendungssystems erlaubt.
Darüberhinaus lassen sich über die in dem der Übertragungsdatei zugrundeliegenden Datenmodell hinaus möglichen Instanzen als sogenannte benutzerdefinierte Instanzen, gekennzeichnet durch ein "!" vor dem Instanznamen, übertragen. Die hiermit ermöglichte Flexibilität der Bildung der selbstdefinierter Instanzen wäre jedoch auch durch die Definition eigener Entitäten auf Modellebene möglich[7].

2.4.2 SDAI als API für STEP-Produktdatenbasen

Mit der in /44/ genormten Zugriffsschnittstelle SDAI (**S**tandard (oder STEP) **D**ata **A**ccess **I**nterface) wird anders als zunächst zu vermuten, nicht die Realisierung einer Verwaltungseinheit für STEP-Daten genormt, sondern eine einheitliche Schnittstelle zu einem System, welches Daten in einer beliebigen Struktur beinhaltet. Als Zugriffsmechanismus wird ein funktionaler (C-late-Binding) bzw. struktureller (C++-Early-Binding) Zugriff auf nach EXPRESS definierten Schemata ermöglicht.
Auch die Definition als Zugriffsschnittstelle ist zumindest mißverständlich; die Spezifikation zielt eindeutig auf die Definition einer API (**A**pplication **P**rocedural **I**nterface) ab. Un-

6. *Das Verfahren des internen Mapping ist für alle nicht komplexen Instanzen für die Konformitätsklasse 1 (CC1) des Parts 21 anzuwenden; die Konformitätsklasse 2 (CC2) verlangt für alle Instanzen externes Mapping.*
7. *Dies gilt strenggenommen jedoch nur für Austauschdateiparser, die die gültige Instanzenmenge aus dem EXPRESS-Modell zur Laufzeit bestimmen.*

ter einem SDAI läßt sich die konkrete Ausbildung des Verwaltungssystems beliebig verstecken wie in Abb. 9 gezeigt wird.

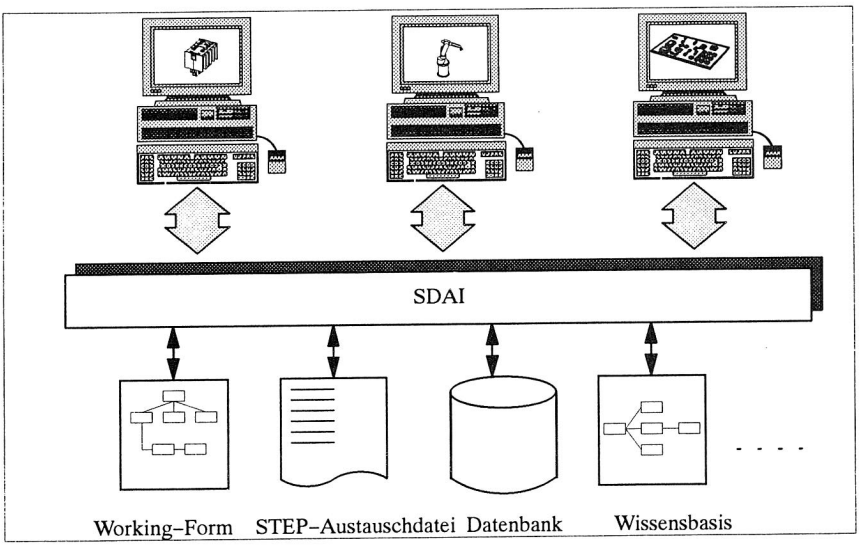


Abb. 9 Realisierung eines SDAI über verschiedenen Basistechnologien

Mit dem 'Standard Data Acess Interface' (SDAI) soll eine Zugriffsschnittstelle für Anwendungssysteme definiert werden, die auf STEP-konforme Daten zugreifen sollen. Die Implementierungstechnik wird dabei oft als 'data sharing' bezeichnet. Hierbei ist der Begriff des 'sharing' allerdings sehr eingeschränkt zu verstehen. Ein Teilen von Anwendungsdaten für mehrere gleichzeitige Anwendungsprozesse und die damit einhergehenden Folgen wie 'Concurrency Control' und 'Lock Management' werden nicht unterstützt. Das SDAI ist dabei im wesentlichen als API (Application Procedural Interface) konzipiert. Die Begriffsdeutung als Zugriffsschnittstelle muß damit erläutert werden. Probleme bzw. Einschränkungen die diese Konzeption im Umfeld der Integration von CAD-Systemen mit sich bringt, wird weiter in Kapitel 5.3.2 verdeutlicht.

Das SDAI definiert zwei Arten von persistenten Daten innerhalb eines Repositories. Dies sind zum einen die Anwendungsdaten, die von einem EXPRESS Model instantiierten Entitäten, und zum anderen ein SDAI 'Data Dictionary'. Das SDAI Data Dictionary stellt im wesentlichen ein Modell von EXPRESS Modellen dar. Dieses ist damit ein EXPRESS Meta Modell, und kann als ein im Datenbankbereich übliches Data Dictionary verstanden werden. Das SDAI Data Dictionary erlaubt über dem SDAI nur lesende Zugriffe und dient Anwendungssystemen zur Information über die Struktur eines EXPRESS Schemas, das die Daten definiert, die dieses Anwendungssystem über das SDAI benutzen kann.

Die Spezifikationen der Sprachanbindungen an das SDAI unterscheiden ein frühes (*early binding*) und ein spätes Binden (*late binding*). Eine SDAI-Implementierung, die dem Prinzip des frühen Bindens folgt, ist bisher nur für die C++-Sprachanbindung definiert.
Ein spätes Binden definiert eine Funktion zum Setzen eines Wertes 'value' eines Attributs 'attrib' einer Instanz 'inst' eines Entitytyps 'ent' als:

```
SetValue(inst, attrib, value)
```

Die Instanz 'inst' des Entitytyps 'ent' muß hierbei in einer vorausgegangenen Operation bestimmt werden. Eine analoges Ergebnis kann durch eine Funktion mit früher Bindung hingegen als:

```
SetValueentattrib(inst, value)
```

erreicht werden. Während eine Implementierung nach dem späten Bindung schema- und damit anwendungsunabhängig ist, ist eine Implementierung mit frühem Binden schemaabhängig und damit abhängig von einem gewählten Anwendungsprotokoll. Ein frühes Binden jedoch kann eine implementierungstechnisch sicherere und hinsichtlich Performanzeigenschaften günstigere Ergebnisse erreichen als ein spätes Binden. Dies wird in Kapitel 3.3.7 weiter verdeutlicht werden.

Die Spezifikation des SDAI legt hinsichtlich ihrer Implementierung zunächst nicht fest, auf welchem Typ von Basissystem diese zu geschehen hat. Im Zusammenspiel mit der Zugriffsmechanik auf die Datenbestände kann jedoch zunächst ein navigierender Zugriff auf die Datenbestände erkannt werden. Dies läßt erkennen, das ein Basissystem der Implementierung zumindest günstige Rahmenbedingungen für einen navigierenden Zugriff erlauben sollte. Die zusätzlich definierte QUERY Operation des SDAI hingegen stellt darüberhinaus auch einen qualifizierenden Zugriff zur Verfügung.

Die für den realisierten Anwendungsfall von Anwendungssystemen wie CAD-Systemen sinnvolle Funktionalität eines Modell-Check-In bzw. Check-Out wird durch die bisherigen Vorschläge für das SDAI nicht berücksichtigt. Dies hat Einfluß auf die Realisierung von CAD-SDAI-Kopplungen wie sie in Kapitel 5.3.2 beschrieben sind.

2.4.3 Alternativrealisierungen

In /40/ wird neben den bereits angesprochenen und zur Standardisierung anstehenden Implementierungsformen 'Austauschdatei' und SDAI noch weitere Implementierungsformen genannt. Dies sind zum einen Wissensbasen und 'Working Forms'.

Während Implementierungsformen in Wissensbasen in der Literatur nicht erwähnt werden sind Implementierungen als 'Working Forms' eher zu beobachten. 'Working

Forms' sind als Realisierungen von EXPRESS-Modellen zu verstehen, die transient im Hauptspeicher zur Verfügung stehen. Realisierungen solcher 'Working Forms' können für viele Anwendungsfälle interessant sein. Auf sie wird in Kapitel 3.3.1 und 3.3.2 weiter eingegangen.

In /14/ und /190/ wird auf die Implementierungen von STEP-Produktdatenbasen durch CORBA /192/ eingegangen. Diese Implementierungen sind jedoch von ihrer Bedeutung letztendlich wie Implementierungen auf objektorientierten Datenbanken zu behandeln.

2.5 Bewertung von STEP

Hier soll eine kurze Bewertung des Ansatzes bei der Entwicklung bzw. Entstehung von STEP vorgenommen werden. Dies soll vor dem Hintergrund der hier dargestellten, kurzen und daher zwangsläufig unvollständigen Einführung in STEP geschehen. Sie soll letztendlich nur eine Begründung für den zu erwartenden Durchsetzung von STEP in der industriellen Praxis liefern.

STEP ist die erste Standardisierung im Produktdatenbereich, die einem derart methodischen Ansatz unterliegt. Die wesentliche Stärke von STEP liegt in der Trennung zwischen Modellbeschreibung und der Spezifikation von Implementierung von STEP-konformen Anwendungssystemen aus dieser Modellbeschreibung heraus.

Einen wesentlichen Anteil an der vorhandenen Methodik hat die Modellierungssprache EXPRESS, die damit ein fundamentaler Baustein dieser Methodik ist. Die Bedeutung von EXPRESS innerhalb STEP ist derart wichtig, daß in vielen Veröffentlichungen auch von STEP/EXPRESS gesprochen wird. Die Sprache EXPRESS stellt, da sie innerhalb STEP als eigenständige Modellierungssprache normiert wurde, die erste international genormte Modellierungssprache dar. Die Verwendung von EXPRESS als Modellierungssprache über die ISO-10303 hinaus wie z.B. in der ISO-15384 zeigt, daß EXPRESS trotz aller Schwächen weitere Verbreitung finden wird. Wie im Verlauf dieser Dokumentation noch deutlich werden wird, ist davon auszugehen, daß EXPRESS in weiter darüber hinausgehenden Bereichen Anwendungen finden wird. Die Verfügbarkeit EXPRESS-konformer Software, die sogar einer genormten Zertifizierung unterliegen kann, wird diesen Trend verstärken.

Eine weitere Stärke von STEP ist die mögliche Anwendbarkeit in beliebigen Anwendungsbereichen. Die Einbeziehung von Modellen zur Beschreibung elektrischer bzw. elektronischer Produkte ist ein herausragendes Beispiel hierfür. Andererseits ist innerhalb STEP auch die Beschränkung nur auf normierungsrelevante Fragestellungen hervorzuheben. Lösungen von Problembereichen, die in anderen Normen oder Aktivitäten bearbeitet werden, sind in STEP zu benutzen. Beispiele sind die Beschreibung von Zei-

chensätzen (ISO 8859-1), elektrischen Symbolen (IEC 617), Textdokumenten (SGML), Programmiersprachen (ANSI-C) und Normteilen (ISO-13584).

Die Problematik, ein Gesamtmodell für alle produktbeschreibenden Daten zu schaffen, auch wenn dieses auf irgendeine Weise in Teilmodelle modularisiert wird, hat in STEP, auch in Anbetracht der vorhandenen Struktur wie sie z.B. durch die Aufteilung in Basisressourcen, Anwendungsressourcen und Anwendungsprotokolle ersichtlich scheint, nicht zu einer homogenen Modularisierung geführt. Trotz einiger Modularisierungsansätze in der Modellierungsmethodik, wie sie durch EXPRESS ermöglicht sind, werden indes in STEP nicht berücksichtigt und führen damit auch zu nachvollziehbaren Problemen. Die während der Entstehungszeit entstandenen Modularisierungsansätzen, wie sie in den nicht mehr aktuellen Teilmodellen IPIM (**I**ntegrated **P**roduct **I**nformation **M**odel), GDPM (**G**eneric **P**roduct **D**ata **M**odel), FFIM (**F**orm **F**eature **I**nformation **M**odel), **P**roduct **S**tructure **C**onfiguration **M**anagement **M**odel (PSCM) dokumentiert wurden, zeigen, daß gerade die Synthese von Partialmodellen, wenn man die Basisressourcen überhaupt als solche verstehen will, zu einem Gesamtmodell, die wesentliche Problematik in STEP darstellt.

Die Akzeptanz der Basisressourcen als solche begründet sich in der mathematisch fundierten Beschreibung der Geometriemodelle. Dies wird verständlich wenn man sich verdeutlicht, daß die Geometriemodelle, die in Anwendungssystemen wie z.B. CAD-Systemen benutzt werden, weitgehend definitorisch stabil, allgemein anerkannt, effizient implementiert sind und daher für Modellersteller transparent bleiben. Die darüber hinausgehenden Modelle der Basisressourcen, die etwa Stücklisten oder Material beschreiben, unterliegen viel häufiger den unternehmenseigenen oder branchenspezifischen Randbedingungen und sind daher wesentlich schwieriger allgemeingültig zu beschreiben. Darüberhinaus bleibt fraglich, ob diese Informationseinheiten überhaupt anwendungskontextfrei, wie dies in den Basisressourcen angestrebt wird, beschreibbar sind. Vermutlich aus auch diesem Grund ist die geplante Beschreibung von Form Features aus den Basismodellen herausgelöst worden.

Wesentliche Kritik an STEP ist oftmals aus der Anwendersicht dadurch gekennzeichnet, daß die Entwicklung von STEP über einen Zeitraum, von der Initiierung 1986 bis zur Veröffentlichung der ersten Teile im September 1994, etwa 8 Jahre gedauert hat. Die Verfügbarkeit zertifizierter Prozessoren ist dabei noch nicht eingeschlossen. Sieht man jedoch die Komplexität der Aufgabe und die erreichten Ergebnisse an, so bleibt festzuhalten, daß offensichtlich auch dieser Zeitraum noch nicht ausreichend bemessen war, um alle Probleme umfassend zu lösen. Das Prinzip, eine modulare Entwicklung und Veröffentlichung von Teilen der Norm vorzunehmen ist dabei nur dann zulässig, wenn ein sauberer Ansatz entwickelt worden wäre, die methodische Integration der Teilmodule zu erlauben. Gerade die Problematik der Interoperabilität der Anwendungsprotokolle zeigt jedoch, daß hier noch Handlungsbedarf besteht. Unter der Randbedingung bereits veröffentlichter Teile der Norm ist hierbei jedoch eine methodische Lösungsfindung ex ante erschwert.

3 Realisierungsumgebungen für STEP

Die STEP-Modularisierungsmethodik gibt eine Trennung zwischen grammatikalischer Definition der Modellierungssprache EXPRESS, der anwendungsunabhängigen Definition von Teilmodellen in den Partialmodellen und der anwendungsorientierten Definition von Anwendungsmodellen in den Anwendungsprotokollen vor. Diesen Modellierungsebenen lassen sich Systeme für die sprachorientierte Verarbeitung von Modellen im EXPRESS-Sprachraum einerseits, der anwendungsfreien Partialmodellverarbeitung andererseits und der Bildung von Anwendungssystemen zuordnen (siehe Abb. 10)[8].

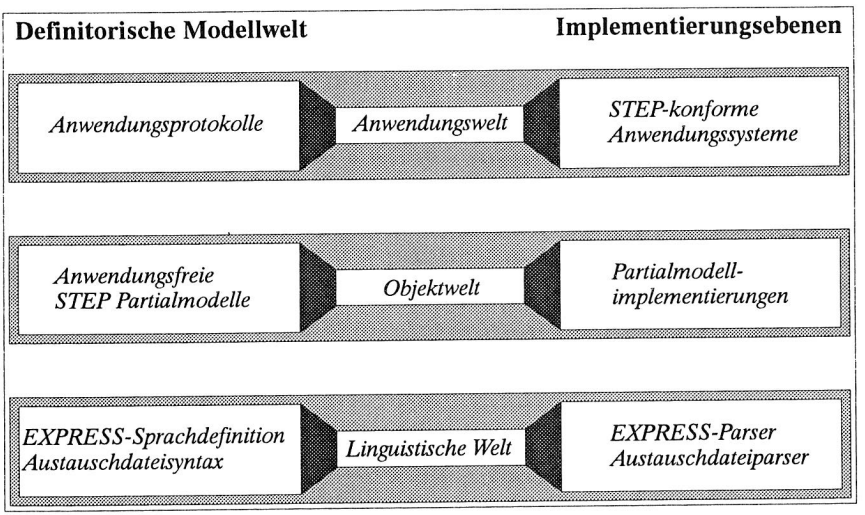


Abb. 10 *Trennung von sprachorientierten Basissystemen und anwendungsorientierten Anwendungssystemen in der STEP Entwicklungsmethodik*

Diese logische Trennung ist eine wesentliche Voraussetzung für eine effektive Implementierbarkeit von STEP-konformen Softwaresystemen wie dies im folgenden dargestellt wird.

In diesem Kapitel soll die Technologie der Systementwicklung sowohl für sprachorientierte Basissysteme als auch für Anwendungssysteme erläutert werden. Dabei wird im wesentlichen auf das für die vorliegende Arbeit wichtige Basissystem aus EXPRESS-Parser und STEP-Austauschdatei-Parser eingegangen. Darauf aufbauend wurde eine Entwicklungsumgebung für STEP-Anwendungen in einer CASE-Umgebung realisiert. Auf die Realisierung von STEP-Datenbanken wird dann im Kapitel 4 sowie auf Anbin-

8. *Der Begriff der Anwendungsorientierung mag im Zusammenhang mit Prozessoren sowie Datenbankschnittstellen unzutreffend erscheinen, die in den STEP APs vorgenommene Modellbildung ist hingegen eindeutig auf Anwendungsmodelle hin orientiert.*

dungen von CAD-Systemen auf STEP-Datenbasen in Kapitel 5 eingegangen. Die Darstellung der Realisierung von STEP-konformen Anwendungssystemen ist dann Kapitel 7 vorbehalten.

3.1 STEP als Entwicklungsumgebung für Ingenieuranwendungen

Softwaresysteme im Ingenieurbereich, die geometrische (Produkt-)Modelle manipulieren, basieren zumeist auf Datenmodellen, die explizit für die jeweilige Anwendung spezifiziert wurden. Die unterschiedlichen Anforderungen, die die Grundlage dieser Modelle bilden, führen damit oft zu hochspezialisierten, wenig allgemeingültigen Produktmodellen. Beispiele hierfür sind 2D-Modelle für alle Arten zeichnungsorientierter Produktmodelle oder höhenattributierte 2D-Modelle für die Blechverarbeitung. Die hohe Spezialisierung der Modelle verhindert dabei die Anbindung bzw. Integration in größere Umgebungen, bzw. die Nutzung der Modelle in bei der Spezifikation nicht geplanten Anwendungsbereichen. Die Spezifikation derartiger Produktmodelle ist oftmals auch von einem uneinheitlichen Verständnis für das zu beschreibende Objekt getragen, und führt damit auch zu inkompatiblen Modellen. Das meist problemorientierte Verständnis der Modelle induziert oft effiziente Hauptspeichermodelle, die meist jedoch nur unzureichend auf Massenspeicher, z.B. Dateien oder Datenbanken abgebildet werden.

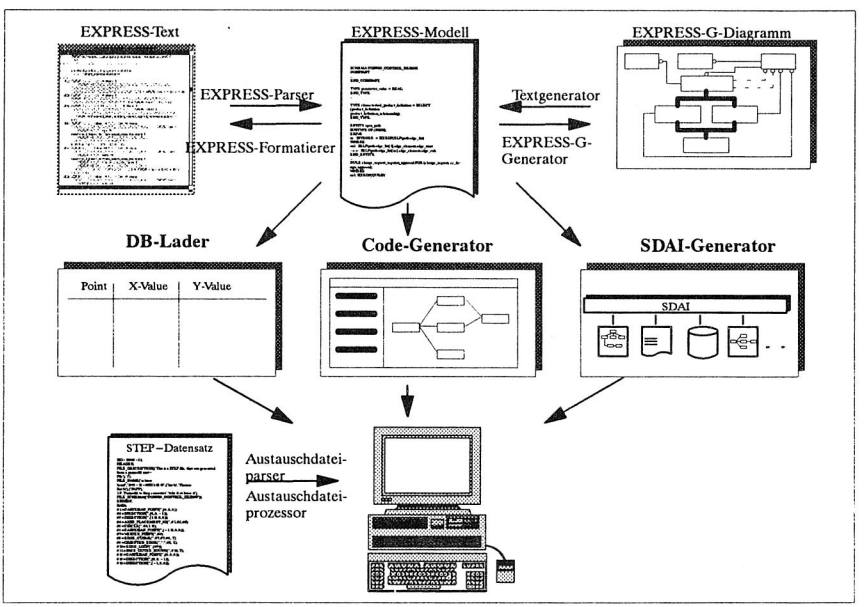


Abb. 11 Realisierungsaufgaben und -werkzeuge für STEP-konforme Anwendungen

Durch STEP werden prinzipiell alle für die Produktdatenverarbeitung relevanten Informationen zunächst anwendungsunabhängig in den Basismodellen, bzw. darauf aufbauend in den AICs und anwendungsorientierten Basismodellen bereitgestellt. Über die in den APs standardisierten Anwendungsmodelle und den daraus entstehenden Anwendungssystemen hinaus, werden unternehmensintern Spezialprobleme, die in den standardisierten Anwendungsmodellen nicht behandelt werden, unter Benutzung der Basismodelle, AICs und anwendungsorientierten Basismodellen Anwendungen entwickelt werden. Die Benutzung der Basismodelle entspricht der im Ingenieurwesen der Synthetisierung von komplexen Systemen aus Modulen und unterstützt damit die im Softwareentwicklungsbereich oftmals geforderte Wiederverwendung von Softwaresystemen. Ob dies einer Objektorientierung nach /89/ entspricht, muß dabei weiter untersucht werden.

Das Defizit von allgemein anerkannten Modellierungs- und Implementierungsmethoden im technischen Bereich wird erst durch den Vergleich mit den Methoden kommerzieller Anwendungssysteme deutlich. Als Basis kommerzieller Anwendungssysteme haben sich seit den siebziger Jahren das relationale Modell durchgesetzt. Die mathematisch fundierte Definition durch das Relationenkalkül sowie die von Implementierungen abstrahierende standardisierte Sprache SQL hat die Grundlage für Entwicklungsumgebungen geschaffen, die im kommerziellen Bereich einen effektiven Standard geschaffen haben. Mit darauf basierenden Modellierungsmethoden, wie der Entity-Relationship Entwicklungsmethode wurde die Lücke zwischen der Modellierung von Anwendungssystemen und deren Implementierung geschlossen. Mit der Methodik, die mit der Entwicklung von STEP einhergeht, wird für den technischen Bereich erstmals ein adäquates Modellierungsprinzip erreicht.

3.2 Entwicklungswerkzeuge für EXPRESS

Die Sprache EXPRESS stellt für die Entwicklung von STEP die fundamentale Grundlage einer Sprache für die Bildung von Modellen bereit, die ein allgemeines Verständnis von Sachverhalten, die von Interesse für ein Anwendungssystem sind, bereit. Als Basis für die Implementierungsmethoden, die formal aus den EXPRESS-Modellen abgeleitet werden, stellt EXPRESS den wichtigsten Teilbereich der formalen Grundlagen von STEP bereit. Anwendungssysteme müssen prinzipiell in der Lage sein, die formale Abbildung von EXPRESS-Modellen in eine Implementierung zumindest abstrakt nachzubilden. Damit kommt der Verarbeitung der Sprache EXPRESS eine grundlegende Bedeutung in STEP zu.

3.2.1 EXPRESS-Compiler

Werkzeuge für die rechnergestützte Verarbeitung formaler Sprachen werden Compiler /135//136/ genannt. Ein Compiler kann im wesentlichen in die Bausteine:

- Scanner
- Parser
- Semantik-Analyse
- Zwischencode-Generator
- Optimierer und
- Code-Generierer

modularisiert werden. Während der Scanner für die lexikalische Analyse zuständig ist, und die Aufteilung eines Eingabestroms in logische Einheiten vornimmt, überprüft ein Parser die syntaktische Korrektheit der Folge der erkannten Einheiten und erstellt gegebenenfalls ein Modell des erkannten Eingabetextes (Parse Tree). Der Optimierer kann die von dem Parser aufgebaut Struktur der Eingabe hinsichtlich gewisser Kriterien optimieren. Der Code-Generator erzeugt aus der Struktur den gewünschten Output. Für die Implementierung der Teile Scanner und Parser existieren eine Reihe leistungsfähiger Werkzeuge, die meist auf einer formalen Spezifikation der Sprache beruhen. Die bekanntesten hiervon sind die im UNIX-Bereich weitverbreiteten Werkzeuge *lex* (**lex**ical analyzer) für die lexikalische Analyse sowie *yacc* (**y**et **a**nother **c**ompiler **c**ompiler) für die syntaktische Analyse. Diese Werkzeuge beruhen auf den fundamentalen Prinzipien eines deterministischen endlichen Automaten (DFA, Deterministic Finite Automata) im Falle von lex bzw. auf dem Prinzip der Kellerspeichers (Stack) im Falle von yacc. Die weitere Beschreibung der Realisierung eines EXPRESS-Parsers folgt in Kapitel 3.2.2.

Die zum aktuellen Zeitpunkt existierenden EXPRESS-Parser bzw. Compiler sind zumeist in umfassendere Systemumgebungen integriert. Dabei wird im oftmals auf die Anforderungen aus der Systemumgebung Rücksicht genommen, und zumeist auf eine umfassende Modellanalyse verzichtet. Die Orientierung von Implementierungskonzepten auf die wesentlichen Elemente von EXPRESS wie Entitäten und Typen lassen komplexere Einheiten der Sprache wie algorithmische Konstrukte und viele semantische Charakteristika der Sprache meist unberücksichtigt.

Die in /41/ vorgenommene Einteilung von Implementierungsniveaus für EXPRESS-Parser charakterisiert folgende vier Ebenen:

Evene 1: Referenzüberprüfung: Der Compiler sollte in der Lage sein, eine komplette Syntaxüberprüfung sowie eine Referenzüberprüfung der in den Modellen definierten Symbole vornehmen zu können.

Ebene 2: Typüberprüfung: Die Typüberprüfung muß die Typkompatibilität in Ausdrükken und Zuweisungen sicherstellen.

Ebene 3: Wertüberprüfung: Die Ebene drei fordert die Überprüfung der Einhaltung von Werteinschränkungen, die in der formalen Spezifikation angegeben sind.

Ebene 4: Komplettüberprüfung: Diese Ebene beinhaltet die Überprüfung aller Anweisungen, die eine Konformität eines Modells beinhalten.

Jede Ebe beinhaltet die Einhaltung aller niedrigerer Ebenen. Über diese Qualitätseinteilung hinaus sind in den Implementierungshinweisen (Protocol Implementation Conformance Statement, PICS) Hinweise zu Implementierungseinschränkungen zu geben.

Die Anforderung aus der Konformität der EXPRESS-Compiler fordert in Ebene 4 die Überprüfung eines Modells auf alle im EXPRESS 'Language Reference Manual' (LRM) definierten Eigenschaften. In dieser Forderung wird offensichtlich unbewußt Eigenschaften eines EXPRESS-Parsers gefordert, die nachweislich nicht erbracht werden können. So ist hierbei das in der Literatur als 'Halteproblem' bekannte, nicht lösbare Problem der Nachweisbarkeit des Endens eines Algorithmus gefordert.

Trotz dieser Information ist die Leistungsfähigkeit eines EXPRESS-Parsers aus den *PICS* nicht vollständig zu erfassen. Eigenschaften, die die praktische Einsetzbarkeit der EXPRESS–Parsers beeinflussen, sind nur durch Analysen festzustellen. In der folgenden Tabelle sind die aktuell bekannten EXPRESS-Parser sowie deren Charakteristika aufgeführt, wie sie aus Produktinformationen zu extrahieren sind.

Name	**Hersteller**	**Scanner-/ Parser-generatoren**	**Level**	**Schema-referen-zauflö-sung**	**instan-tiierbares Modell ?**	**Code-Generie-rung/ Ausgabemodule**
fedex	*NIST*	*lex/flex yacc/bison*	*1*	*Parsen*	*ja*	*programmierbar/ C++, SQL EXPRESS*
Expparse	*FAPS*	*flex yacc/bison*	*4*	*Parsen*	*ja*	*wie oben zusätzlich Postgres/ Kappa*
ice	*CADLAB*	*ELI*	*4*	*Parsen, in-nerhalb Datei*	*nein*	*C++ geplant*
PRODEX-compiler	*PRODEX-Consortium*	*?*	*?*	*?*	*ja*	*programmierbar*
EXPRESS-LAB	*GOPAS*	*?*	*?*	*?*	*?*	*ONTOS-Schema*
expparse	*STEPtools*	*?*	*4*	*?*	*?*	*ROSElib*
	UniSQL	*?*	*?*	*?*	*?*	*UniSQL-Schema*
	ITI	*?*				
ProSTEP Toolkit	*ProSTEP/ GIDA*	*Referenzta-bellen*	*?*	*Binden*	*über CDA*	-
ECCO	*RPK*	*Rex*	*?*	*Parsen, in-nerhalb Datei*	*ja, über ge-nerierte C++ An-wendung*	*C++*
ex	*RAL*	*Referenzta-bellen*	*?*	*?*	*ja*	–

Tab. 1 Charakteristika von EXPRESS-Parsern/Compiler (u.a. aus /76/)

Die Bezeichnung EXPRESS-Parser, wie sie hier zumeist benutzt wurde bezieht sich im wesentlichen auf die Tatsache, daß hier zunächst nicht die Fähigkeit der Code-Generierung im Vordergrund steht. Da jedoch die in STEP definierten Implementierungstechniken hierauf basieren, bieten EXPRESS-Parser zumeist entweder die Möglichkeit, ein für STEP-Implementierungen eigens zu realisierendes Ausgabemodul für den Parser zu realisieren, oder der Hersteller liefert solche Ausgabemodule für dedizierte Implementierungssysteme mit aus.

Da die in EXPRESS formulierbaren Modelle nur Schemata für die Instanzen eines Modells darstellen, diese Instanzen aber in STEP-Austauschdateien dargestellt werden müssen, ist es von Interesse, ob ein durch einen EXPRESS-Parser erzeugtes Modell auch Instanzen dieses Modells verarbeiten kann (siehe Tab. 1). Dies ist eine Voraussetzung für die in Kapitel 3.3.2 dargestellte Technologie der generischen Austauschdateiparser.

Die Problematik der Auflösung von Referenzen von Elementen innerhalb eines Schemas auf ein externes Schema (USE FROM, REFERENCE FROM) stellt ein wichtiges Implementierungscharakteristika eines EXPRESS-Parsers dar. Im wesentlichen lassen sich hierfür zwei Strategien unterscheiden:

a) Getrenntes Parsen der einzelnen Modelle und nachträgliches Binden der nicht aufgelösten Symbole und

b) Auflösen der referenzierten Symbole während des Parsevorganges durch erzwungenes Einfügen der referenzierten Schemata.

Während für die Lösung a) ein einfacheres Verwaltung einzelner Schemata sowie ein schnellerer Parsevorgang spricht, ist aus modellierungstechnischer Sicht Strategie b) vorzuziehen, da ein korrekter Parsevorgang im Falle a) nicht notwendigerweise ein korrektes Modell impliziert. Ein wesentlicher Teil des Parsevorganges, nämlich die Typprüfung von referenzierten Elementen wird dabei in den Bindevorgang verlegt. Dies bedeutet, daß während einer Modellentwicklung Fehler spät erkannt und damit verschleppt werden können. Strategie b) hingegen kann nach Beendigung des Parsevorganges sicher ein korrektes bzw. fehlerhaftes Modell identifizieren.

3.2.2 Das NIST STEP-Toolkit

Am NIST (**N**ational **I**nstitute for **S**tandardization and **T**echnology) wurde im Rahmen früher Evaluierungsuntersuchungen von STEP-Implementierungen ein Satz von prototypischen Softwarewerkzeugen entwickelt. Dieser Satz von Werkzeugen beinhaltet einen EXPRESS-Parser (*Fed-x*), einen STEP-Austauschdatei-Parser (P21), Ausgabemodulen für den EXPRESS-Parser, namentlich einen C++-Klassengenerator (*fedex++*), einen EXPRESS-Textformatierer (EXPRESS-Pretty Printer, exppp), sowie einer C++ Klassen-

bibliothek zur Handhabung von Instanzen von EXPRESS-Entitäten. Aufbauend auf diesen Tools existiert ein System, mit dessen Hilfe eine Benutzeroberfläche zum Editieren von STEP-Austauschdateien über einem prototypischen SDAI (DataProbe) implementiert wurde.

Dieses Toolkit, das vom NIST als Quellcode ohne Einschränkung der Berechtigungen veröffentlicht wurde, ist für die hier vorliegende Arbeit in wesentlichen Teilen weiterentwickelt worden und dient als Basis eigener STEP-Implementierungen (siehe Kapitel 3.3).

Der EXPRESS-Parser Fed-x

Das Konzept des Fed-x (**Fed**eral E**x**press Compiler) basiert auf dem konventionellen mehrstufigen Parse-Verfahren, wie es z.B. in /135/, /136/, /137/, /74/ dargestellt wird.

Als Anwendungsbeispiel für die 'Code-Generierung' des EXPRESS-Parsers *Fed-x* sind vom NIST Programmodule realisiert worden, die den Parse-Tree des *Fed-x* als zu traversierende Struktur nutzen, um einen spezifischen Output zu erzeugen.

Die Einschränkung der Funktionalität des *Fed-x* auf die Module Lexikalische Analyse, Syntaktische Analyse, Semantische Analyse und Zwischencode-Generierung nach Abb. 12 stellt einen wesentlichen Teil des Konzeptes des NIST-STEP-Toolkits dar. Die für beliebige Abbildungen von EXPRESS auf Anwendungsimplementierungen nötige Generierung eines Codes läßt sich durch Realisierung eines sogenannten 'backends' des *Fed-x* vornehmen. Dieses Backend muß den innerhalb eines Prozeßraumes erzeugten Zwischencode (Parse-Tree) des *Fed-x*, die sogenannten EXPRESS–Working-Form, in einen gewünschten Code umsetzen. Beispiele hierfür sind später in diesem Kapitel oder in Kapitel 3.3.4 zu finden.

Der STEP-Austauschdatei-Parser P21

Der Parser zum Lesen von STEP-Austauschdateien, *P21*[9] ist nur als eine zweite Stufe des *Fed-x* anwendbar: Die für den Aufbau der Parse-Trees des STEP-Austauschdatei notwendige Basisstruktur wird von *Fed-x* bereitgestellt. Die Instanzen aus der Austauschdatei werden als abhängige Strukturen der Strukturen der Entitäten aus dem EXPRESS-Schema eingefügt (Abb. 16) und ergeben dann die sogenannte STEP-Working-Form.

Der EXPRESS-Modell Formatierer exppp

Ein Beispiel für ein Ausgabemodul des *Fed-x* wurde in einem Formatierungsmodul für die geparsten EXPRESS-Modelle der 'EXPRESS Pretty Printer' *exppp* realisiert.
Neben der Fähigkeit, einen beliebig gestalteten korrekten EXPRESS-Eingabetext in eine für einen menschlichen Betrachter lesbare Form zu bringen, kann mit diesem Modul nachgewiesen werden, daß *Fed-x* in der Lage ist, ein korrektes EXPRESS-Modell in

9. *Teil 21 der ISO-10303 beschreibt das Format der STEP-Austauschdatei.*

einen internen Parse-Tree zu überführen. Durch Identität der eingelesenen mit den geschriebenen Modellen kann geschlossen werden, daß der Parse-Tree ein semantisch vollständiges Abbild des Eingabetextes darstellt.

Die STEP Class Library/DataProbe

Das *DataProbe* stellt einen Editor für Instanzen von EXPRESS-Schemata zur Verfügung. Es ist in der Lage eine STEP-Austauschdatei zu lesen, die darin enthaltenen Instanzen zu verändern, zu löschen oder Instanzen zu erzeugen. Die Instanzen können dann wiederum in einer STEP-Austauschdatei gespeichert werden. Der Zugriff auf die Instanzen wird dabei über eine vereinfachte Version eines SDAI /44/ ermöglicht. Damit wird ein SDAI über Dateien als Produktdatenbasen realisiert (siehe Abb. 9). Die schemaabhängigen Teile des SDAI werden in der Version eines C++ early-binding mit Hilfe eines *Fed-x*-Backends (*Fed-x++*) erzeugt. Eine C++ Klassenbibliothek (STEP-Class-Library, SCL) realisiert den schemaunabhängigen Teil des SDAI. Die Benutzeroberfläche wurde in dem C++ X-Widgetsatz Interviews realisiert.
DataProbe stellt einen ersten Versuch dar, ein Werkzeug zur Manipulation von materialisierten EXPRESS-Instanzen zur Verfügung zu stellen. Daß dies über einer SDAI-Realisierung geschieht hat dabei nur nebensächliche Bedeutung. Dessen Realisierung ist nur äußerst rudimentär und kann die wesentliche Forderungen an ein SDAI nicht erfüllen.

3.3 Eine Entwicklungsumgebung für STEP-Anwendungen

Die Realisierung von STEP-kompatiblen Anwendungen verlangt die Bereitstellung einer Entwicklungsumgebung, die einerseits die Randbedingungen einhält, die von der STEP-Methodik, bzw. den gewählten Anwendungsprotokollen vorgegeben werden und andererseits die effiziente Implementierung von STEP-konformen Anwendungssystemen ermöglicht. Die für ein derartiges Werkzeugt angestrebte Funktionalität läßt sich wie folgt charakterisieren /149/:

- Parser/Compiler für EXPRESS
- Parser/Compiler für Instanzen (STEP-Austauschdatei/EXPRESS-I)
- Modellierungsumgebung für EXPRESS/EXPRESS-G/EXPRESS-I-G
- Modellierungsmöglichkeit für SADT/IDEF1X/NIAM
- Möglichkeit zur Erzeugung von Instanzen
- Bereitstellung generischer/anwendungsspezifischer STEP-Partialmodelle/Anwendungsprotokolle
- Datenbankanschluß
- SDAI-Generierung (Generisch, konfigurierbar)

- CAD-Anschluß
- Unterstützung der Entwicklung von Benutzeroberflächen
- Visualisierung von geometrischen Objekten, die in den STEP-Geometriemodellen dargestellt werden können,
- Anbindungen an Programmiersprachen
- Validierung von Instanzenmengen hinsichtlich des korrespondierenden EXPRESS-Schemas, insbesondere Auswertung der Konsistenzbedingungen

Um die Entwicklung spezifischer Anwendungen für die STEP-Integration zu unterstützen, wurde im Rahmen der Arbeit eine Entwicklungsumgebung Anwendungssysteme geschaffen, die einen großen Teil der oben erfaßten Anforderungen erfüllt. In den folgenden Kapitel wird diese Umgebung dargestellt.

Die realisierte Entwicklungsumgebung kann als eine Umgebung verstanden werden, wie sie für die Entwicklung von Anwendungssystemen existieren, die auf relationalen Datenbanken basieren. Diese Entwicklungsumgebungen, oftmals basierend auf 4th GL-Sprachen und Werkzeugen zur Informationsmodellierung mit Hilfe des Entity-Relationship-Modells, werden von Anbietern relationaler Datenbanksysteme angeboten und haben im Bereich der Anwendungsentwicklung für kommerzielle DV-Systeme eine hohe Akzeptanz und eine weite Verbreitung erreicht. Das hier vorzustellende Werkzeug stellt in diesem Sinn eine Analogie für die technische Anwendungsentwicklung dar. Durch die in STEP definierte Entwicklungsmethodik eröffnet sich damit die Chance, durch effektive Unterstützung der Anwendungsentwicklung eine ähnlich hohe Akzeptanz in der Softwareentwicklung im technischen Bereich zu erreichen.

Zu Beginn der Entwicklungsarbeiten der vorliegenden Dokumentation existierten auf dem Mark noch keine Entwicklungswerkzeuge dieser Art. Im Laufe der fortschreitenden Konkretisierung der Teile der ISO-10303 sind mehrere derartige Werkzeuge entstanden, die aber oftmals nur einen Teilbereich der geforderten Funktionalität abdecken. Zumeist entstammen sie als Sekundärprodukte Entwicklungen, die von Softwareanbietern vorangetrieben wurden, um eigene STEP-Implementierungen zu ermöglichen.

Name	**Hersteller**	**Module**
NIST-Toolkit	*NIST*	*EXPRESS-Parser, Übertragungsdatei-Parser, EXPRESS-Formatierer, C++-Klassengenerator, C++-Klassenbibliothek, Instanzeneditor, Langformatgenerator, AP-Entwicklungstool*
PRODEX-Toolkit	*Prodex-Konsortium*	*EXPRESS-Parser, Übertragungsdatei-Parser, Übertragungsdatei-Formatierer, SDAI mit C-Binding*
PISA-Toolkit	*PISA-Konsortium*	*EXPRESS-Parser, Übertragungsdatei-Parser, Austauschdateiparser*
Cas.CADE	*Matra Datavision*	*EXPRESS-Parser, Übertragungsdatei-Parser, C++-Klassenbibliothek*
STEPTools Toolkit	*STEPTools*	*EXPRESS-Parser, Übertragungsdatei-Parser, Datenbanklader (ORACLE, ObjectStore, ...), EXPRESS-G-Editor, Prozessoren für ACIS, AutoCAD, u.a., C++-Klassenbibliothek*

Name	Hersteller	Module
ProSTEP Toolkit/ GIDA Toolkit	*GIDA*	*EXPRESS-Parser, EXPRESS-G-Editor, Übertragungsdatei-Parser, Übertragungsdatei-Formatierer, SDAI mit C-Binding, Datenbanklader (ORACLE, u.a)*
EXPRESSLab	*GOPAS*	*EXPRESS-Parser, Übertragungsdatei-Parser, ONTOS-Datenbanklader*
FAPS-Toolkit	*FAPS*	*EXPRESS-Parser, Übertragungsdatei-Parser, EXPRESS-Formatierer, Übertragungsdatei-Formatierer, EXPRESS/STEP-Entwicklungsumgebung, Postgres-Datenbanklader, Geometriemodellviewer, EXPRESS-Editor, Prozessoren für ACIS und Parasolid*
EDM - EXPRESS Data Manager	*EPM Technology*	*EXPRESS-Parser, Übertragungsdatei-Parser, SDAI mit C-Binding*

Tab. 2 Kommerziell angebotene Toolkits zur Unterstützung von STEP-Entwicklungen (u.a. nach /76/)

Die in Tab. 2 aufgelisteten Werkzeuge stellen zumeist einen Teilausschnitt der für umfassende Entwicklungen nötigen Module bereit. Obligatorisch sind jedoch für alle Toolkits EXPRESS-Parser und Übertragungsdatei-Parser. Unterschiedliche Ansätze zur Zwischenspeicherung eines EXPRESS-Modells als Hauptspeichermodell (NIST, PRODEX, PISA) oder als Zwischendatei (ProSTEP, STEPTools) stellen dabei die wesentlichen Architekturprinzipien dar. Weiterhin ist die Anzahl der Datenbanklader auffällig, die meist mit einem Generator für Klassen objektorientierter Programmiersprachen einhergeht. Dies ist zu einem Zeitpunkt, zu dem die SDAI – Spezifikation in einem noch vorläufigen Dokument vorliegt zunächst unverständlich. Dies kann nur aus dem Bedarf nach Übersetzung von EXPRESS Schemata in eine Laufzeitumgebung verständlich werden.

3.3.1 Parsen von EXPRESS-Quelltexten

Das Vorliegen einer formalen Spezifikation der Sprache EXPRESS /41/ legt die Vermutung nahe, Werkzeuge zur Generierung der Teile Scanner und Parser des Compilers durch Compiler-Compiler automatisieren zu können. Die zur formalen Definition von Compiler-Compilern nötigen Eigenschaften einer Sprache sind vergleichsweise restriktiv /136/. Die Definition von EXPRESS erfüllt jedoch einige Voraussetzungen, die die Nutzung von Compilergeneratoren ermöglicht.

Zur Definition einer Scanner-Definition, die die automatische Erzeugung eines Scanners mit Hilfe des weit verbreiteten Scanner-Generators lex /74/,/137/ bzw. flex ermöglicht, muß ein vom Scanner zu erkennendes Muster durch einen regulären Ausdruck beschreibbar sein /137/. Diese Forderung wird durch die Formulierung der lexikalischen Elemente in /41/ erfüllt.

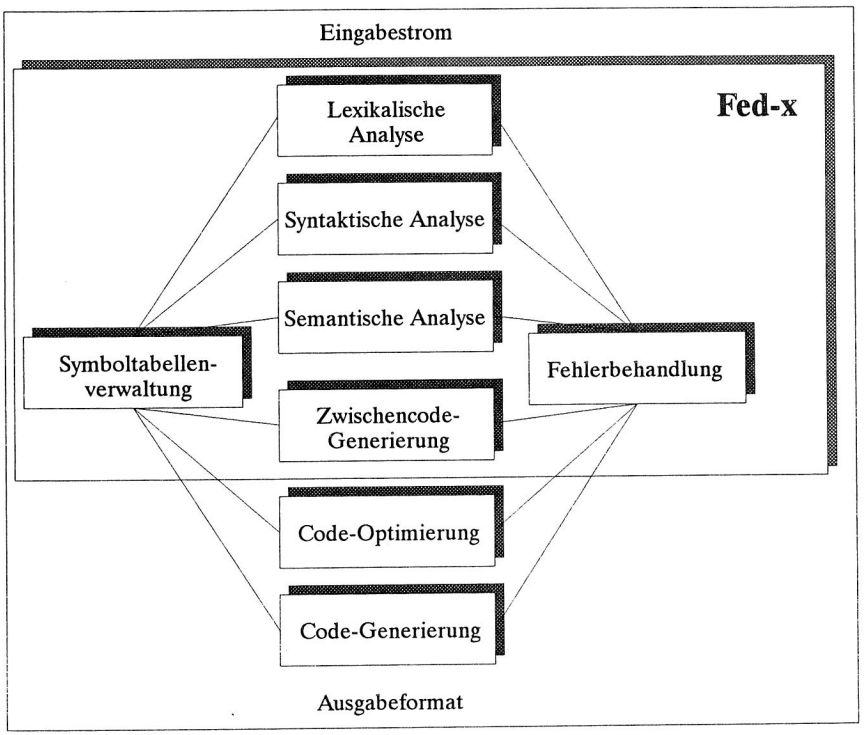


Abb. 12 Unterstützte Komponenten eines Compilers durch 'Fed-x'

Die irrtümliche Annahme, EXPRESS sei eine Sprache, die durch eine kontextfreie Grammatik nach /41/ beschrieben ist /82/, abstrahiert die Tatsache, daß nicht alle Produktionen dieser Grammatik eine gültige EXPRESS-Formulierung darstellen. Die mit Hilfe der in der Wirth-Syntax-Notation (WSN) beschriebene Grammatik ermöglicht zunächst nur die Definition von Produktionen von denen eine Teilmenge gültige EXPRESS-Texte darstellen. Nach /135/ ist eine kontextfreie Grammatik prinzipiell nicht in der Lage, eine Darstellung einer Programmiersprache zu ermöglichen. Jedoch kann eine solche Grammatik einen großen Teil einer Sprachbeschreibung spezifizieren. Anteile der Syntax, die durch diese Grammatik nicht beschreibbar sind, müssen durch nicht formale, oftmals deskriptive Teile in der Spezifikation abgedeckt werden.

Durch die in /41/ vorliegende Grammatik, bzw. der aus ihr abzuleitenden vereinfachten Backus-Naur-Form (BNF) läßt sich eine Eingabebeschreibung für das Werkzeug yacc (**y**et **a**nother **c**ompiler **c**ompiler) oder dessen Derivat bison formulieren. Mit Hilfe von yacc/bison läßt sich ein sogenannter LARL(1)-Parser (**L**ook-**A**head, **L**eft-to-right, **R**ightmost derivation) erzeugen. Dieser Parser stellt einen deterministischen endlichen Automaten (**D**eterministic **F**inite **A**utomaton, DFA) dar, der für sich allein einen Parser für alle nach der in der Grammatik gültigen Eingabetexte bildet. Die über die in der Grammatik

formulierbaren Spracheigenschaften hinausgehenden Charakteristika werden in Durchläufen verarbeitet, die sich an den Scanner/Parser anschließen:

Die Charakteristika von EXPRESS erzwingen einen für moderne Programmiersprachen unüblichen mehrstufigen Parsevorgang (Abb. 13): Symbole können in EXPRESS vor ihrer Deklaration benutzt werden. Dies erzwingt eine unnötig schwache Grammatikbeschreibung der Sprache für den Parsergenerator, da bei die Darstellung der Sprachgrammatik für den Parsergenerator vereinfachende und verallgemeinernde Annahmen gemacht werden müssen.

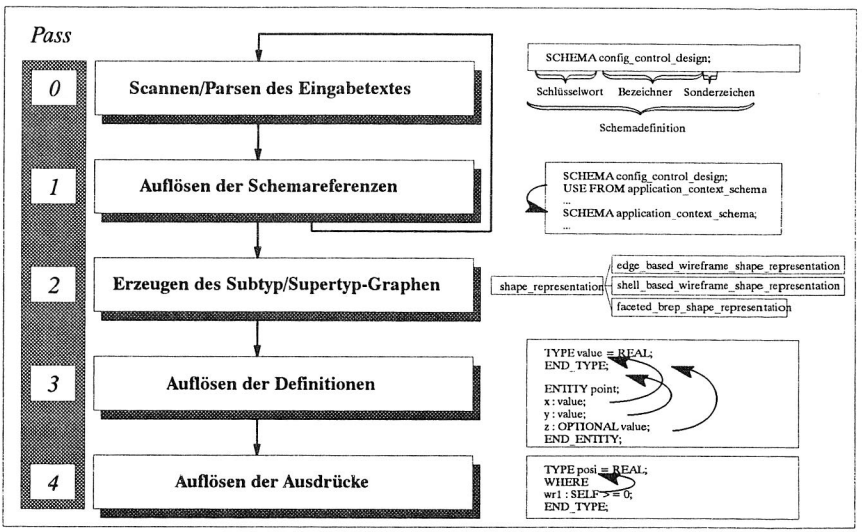


Abb. 13 Funktionale Aufgabenverteilung der Passes des EXPRESS-Parsers

Im Parsevorgang werden die vom Scanner empfangenen Token in einen Syntaxbaum eingetragen. Der zweite Schritt übernimmt die Auflösung der Referenzen von Elementen auf Elemente in anderen Schemata. Das Konzept des Parsers hält sich dabei an das Konzept der erzwungenen Einfügung referenzierter Schemata. Dies ermöglicht die Auflösung von Referenzen zur Parsezeit: Wird auf ein Schema verwiesen, welches sich nicht innerhalb der Eingabedatei befindet, wird versucht, eine Datei mit dem Namen des referenzierten Schemas (erweitert um ein Dateinamensuffix ".exp") zu lesen. Im dritten Schritt wird dann die Bildung des Supertyp/Subtyp-Grafen, im vierten eine Auflösung der (Typ-, Funktions-, Prozedur-, und Regel-) Definitionen und im fünften der Ausdrücke vorgenommen (Abb. 13). Innerhalb der Auflösung der Konstrukte wird dabei jeweils eine Auflösung der innerhalb der Konstrukte benutzten Symbole und die Verbindung der Symbole zu den bezogenen Konstrukten vorgenommen.

Die zum Aufbau des Parse-Trees vorhandenen Basisstrukturen bilden die Konstrukte des Sprache EXPRESS in C-Konstrukten ab. Das grundlegende Element des Parse-

Trees ist der sogenannte Namensraum (*Scope*). Für jedes in EXPRESS definierte Konstrukt, das einen Namensraum umfaßt, wird ein *Scope* erzeugt. Innerhalb dieser *Scopes* sind Symbole eindeutig, d.h. sie stehen für ein innerhalb dieses Namensraumes definiertes Element. Für die EXPRESS-Konstrukte "SCHEMA", "ENTITY", usw. werden derartige *Scopes* generiert. Die innerhalb eines Scopes definierten Symbole werden über eine dynamische Hash-Implementierung in Verzeichnissen (*Dictionary*) gespeichert /6/. Hierdurch sind effiziente Zugriffe auf die Elemente des *Scopes* über ihre Identifikatoren (Namen) möglich. Der Typ eines Namensraumes wird in der Struktur des Scopes (*type*) berücksichtigt.

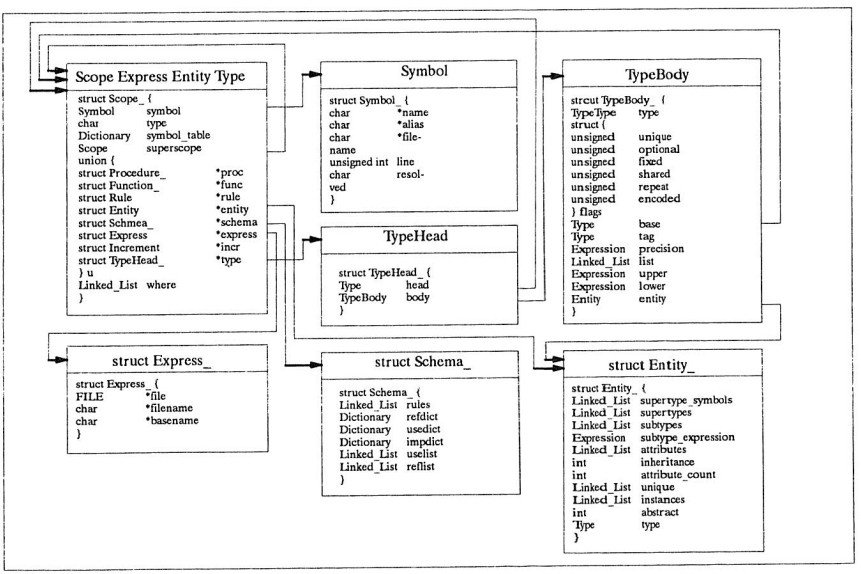


Abb. 14 Teile der durch Expparse erzeugten Working-Form in C-Strukturen

Benannte Typen sowie Typverwendungen[10] werden über die Strukturen TypeHead und TypeBody abgebildet. Die Struktur TypeHead unterscheidet zwischen Typdefinitionen und Typverwendungen. Typdefinitionen werden durch TypeBody wiedergegeben. Der durch den definierten Typ wiedergegebenen Basistyp wird durch eine Enumeration (TypeType) ausgedrückt, die alle in EXPRESS möglichen Typen (siehe Abb. 6) sowie einige für die Verarbeitung nötige interne Typen unterscheidet. In der Struktur *flags* werden die einer Typdefinition zuweisbaren Eigenschaften wie UNIQUE, OPTIONAL und FIXED zugeordnet.

10. *Als Typverwendungen werden Konstrukte bezeichnet, die einen nicht skalaren Typ beschreiben. Beispielsweise kann eine Variable als:*
var : LIST [1:?] OF UNIQUE INTEGER;
definiert werden. Die für die Variable vorgenommene Typdefinition stellt keinen benannten Typ sondern eine in diesem Sinne benutzte Typverwendung dar.

3.3.2 Parsen von STEP-Dateien

Die in /82/ vorgestellte Vorgehensweise zur Generierung anwendungsspezifischer Austauschdateiparser beschreibt folgenden Ansatz: Aus den Schemabeschreibungen wird mit Hilfe eines EXPRESS-Parsers aus einem Schema das Programm (bzw. der Programmcode) eines Parsers für eine Austauschdatei des korrespondierenden Schemas erzeugt. Hierbei wird ein Teil der strukturverarbeitenden Algorithmik innerhalb des Austauschdateiparsers nicht mehr datengesteuert sondern programmgesteuert abgearbeitet. Für den dadurch realisierten Parser bedeutet dies eine etwas höhere Performanz, da das Modellschema nicht mehr für jeden Parsevorgang erzeugt werden muß. Andererseits sind hierbei für jedes Anwendungsprotokoll dedizierte Prozessoren zu generieren.

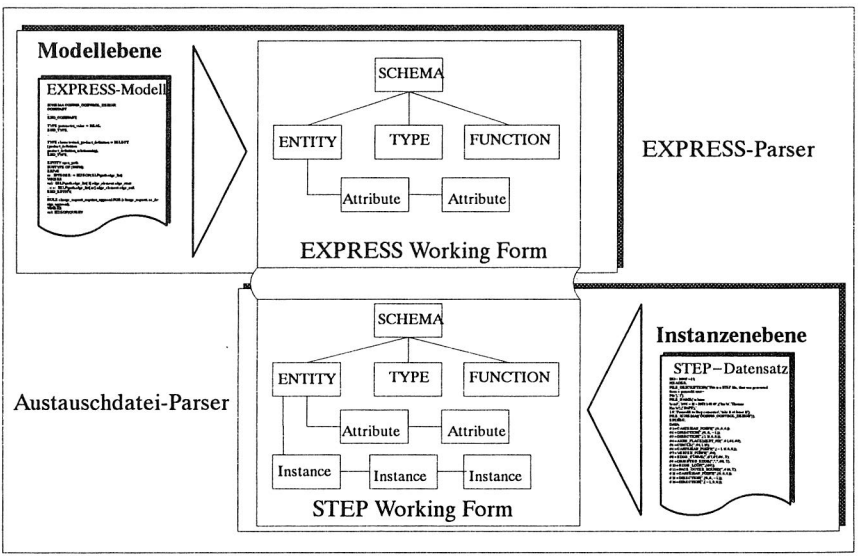


Abb. 15 Konzept des generischen Austauschdateiparsers

Mit dem in Abb. 15 dargestellten Konzept wird die Realisierung eines generischen Parsers für alle Anwendungsprotokolle erreicht. Als notwendige Voraussetzung zum Lesen einer Austauschdatei eines Anwendungsprotokolls ist hierfür lediglich die Verfügbarkeit eines EXPRESS-Schemas des Anwendungsprotokolles. Der wesentliche Vorteil dieses Konzeptes ist die Anwendbarkeit dieses generischen Parsers für alle Austauschdateien für die ein korrespondierendes EXPRESS-Schema zur Verfügung steht. Die für den Aufbau der Schemastruktur nötige Performanzeinbuße muß sich dabei gegenüber der Anwendungsflexibilität, der stärkeren Modularisierung und der logischeren Konzeption rechtfertigen.

Die durch den STEP-Datei-Parser gelesenen Datensätze werden in die vom EXPRESS–Parser erzeugten Datenstrukturen eingefügt. Das EXPRESS Konstrukt ENTITY als pri-

märe in einer STEP-Austauschdatei instantiierbare Einheit besitzt einen Einstiegspunkt für die Instanzen dieser Entität.

Analog zu dem Scope-Konstrukt der EXPRESS Working-Form wurde für die Working Form der STEP-Datensätze (STEP-Working-Form) ebenfalls ein Scope-Konstrukt (P21Scope) eingeführt. Dieses Scope beinhaltet über eine Union sowohl Produkt (struct Product_), die Menge der in einem STEP-Datensatz darstellbaren Instanzen als auch Instanzen (struct Instance_) als solches, die über das SCOPE-Konstrukt selbst eine Menge von Instanzen beinhalten können. Damit wird eine rekursive Anwendung des Scope-Konzeptes möglich.

Alle STEP-Partialmodelle stellen eine Abbildungstabelle zwischen den Typnamen in Langform und Kurzform in einer Liste bereit, die den Kurznamen und Langnamen eines Entitytyps gegenüberstellt. Für die Entitydefinition advanced_brep_shape_representatin aus /61/ wird die Korrespndenz zum Kurznamen 'absr' wie folgt in der Abbildungstabelle definiert:

```
ABSR,ADVANCED_BREP_SHAPE_REPRESENTATION
```

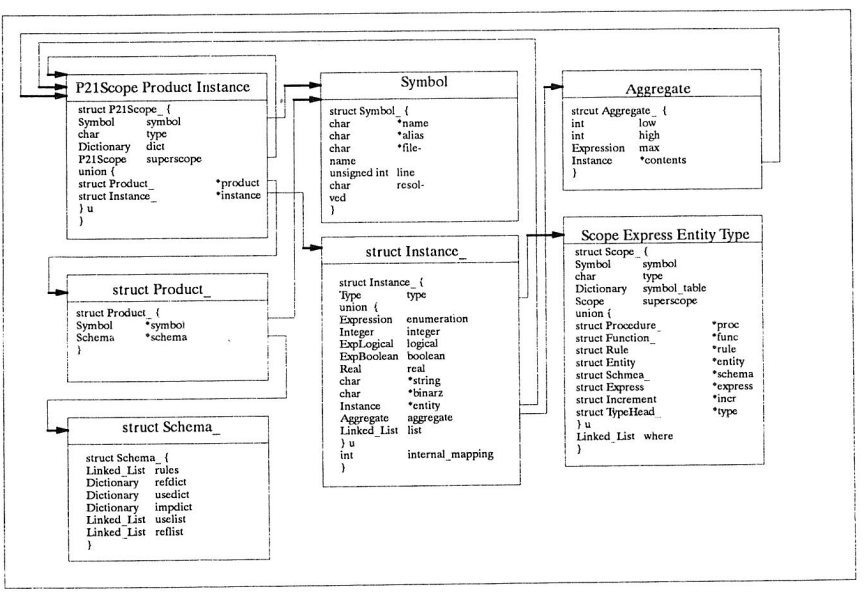


Abb. 16 Ausschnitt aus der STEP-Working-Form

Für die Unterstützung der Kurznamen von EXPRESS-Entitäten werden für alle Symbole dieser Bezeichner ein Alias eingeführt, die den zu einem Typnamen zugeordneten Kurznamen aufnehmen.
Für jedes mit Hilfe des EXPRESS-Schemaparsers geparste Schema wird versucht, eine

wie oben beschriebene Abbildungstabelle zu lesen und darin enthaltenen Kurznamen-Langnamen-Tupel dazu benutzt, den für einen Typnamen angegebenen Kurznamen als Alias in die Symbolbeschreibung aufzunehmen. Damit ist ein problemloses Parsen von Übertragungsdateien möglich, die Instanzen mit Kurznamen enthalten (siehe auch Kapitel 3.3.3).

Die in EXPRESS-Modellen definierten Restriktionen werden oftmals (siehe dazu auch Kapitel 6.1.1) durch lokale (WHERE) oder globale (RULE) Regeln definiert. Die Überprüfung der Einhaltung dieser Restriktionen kann prinzipiell durch zwei softwaretechnisch unterschiedliche Prinzipien erfolgen und zwar durch

- Kompilation der Regeln aus dem EXPRESS-Modell in Objektcode und Einbindung der Kompilats in den Übertragungsdateiparser oder
- Abarbeitung des EXPRESS-Parse-Trees in den Teilen, die für die Einhaltung der Restriktionen verantwortlich sind.

Während der erste Ansatz einen Performanzvorteil verspricht, aber für jedes Modellschema die a-priori Bereitstellung einer eigenen Funktionsbibliothek erfordert, kann der zweite Ansatz flexibler für jedes Modellschema gehandhabt werden.

Für den Austauschdateiparser wurde der zweite Ansatz gewählt der Flexibilitätsvorteil, auch im Hinblick der Handhabbarkeit höher gewichtet wurde als der theoretische Performanznachteil gegenüber dem ersten Ansatz.
Für jede Instanz der in einer Übertragungsdatei enthaltenen Instanzenmenge wird die Entität bestimmt und für alle lokalen Regeln dieser Entität auf alle Instanzen überprüft. Die Überprüfung wird für einige wenige ausgewählte Operatoren (numerisch +,-,*,/) und interne EXPRESS-Funktionen (SIZEOF, EXISTS) unterstützt. Es kann gezeigt werden, daß damit schemaflexibel beliebige Restriktionen auf Instanzen von EXPRESS-Entitäten überprüft werden können.

In /91/ wird die Validierung von STEP-Dateien unter Nutzung paralleler Verarbeitung skizziert. Darunter sollen sowohl statische Typüberprüfungen als auch algorithmisch definierte Restriktionen überprüfbar sein. Hintergrund der Parallelisierung der Verarbeitung wird mit der für reale Anwendungen großen Menge von Instanzen begründet.

Im Gegensatz zu der schemaflexiblen Instanzenüberprüfung im realisiertern Prototypen wird mit Ecco /197/ das Konzept der Constraintüberprüfung durch einen Austauschdateiparser realisiert, der auf einem schemaspezifisch generierten Austauschdateiparser basiert.

3.3.3 Formatieren von Instanzen von EXPRESS Entitäten

Als ein notwendiges Modul zur Generierung von STEP-Datensätzen aus einer hauptspeicherinternen Darstellung heraus ist ein Dateiformatierer realisiert worden, der aus

einer STEP-Working-Form eine STEP-Datei erzeugt. Wird mit Hilfe der Funktionen zur Erzeugung von STEP-Instanzen hauptspeicherintern eine STEP-Working-Form erzeugt, kann mit Hilfe dieses Moduls eine nach ISO-10303-21-konforme Datei erzeugt werden. Damit läßt sich der Vorgang der Instanzenbildung von der Dateigenerierung trennen. Gründe für die funktionale Trennung liegt einerseits in der Notwendigkeit begründet, während der Instanzgenerierung Vorwärtsreferenzen auf später zu erzeugende Instanzen vornehmen zu müssen, und diese Referenzen im Nachhinein auflösen zu müssen. Dies ist bei sofortigem Erzeugen einer Datei nur schwer möglich.

Weiterhin kann durch die funktionale Trennung von Instanzgenerierung und Dateierzeugung das Format der STEP-Austauschdatei unabhängig von der Art der Instanzgenerierung gehalten werden. Die drei unterschiedlichen STEP-Dateiformatprinzipien sind:

- Konformitätsklasse der Austauschdatei
  STEP-Dateien können in einer Konformitätsklasse 1 oder 2 geschrieben werden. In der Konformitätsklasse 1 werden alle Instanzen, die ein Leaf-Node des Suptyp/Supertyp-Graphen darstellen mit internem Mapping geschrieben. Das Verfahren des internen Mapping erzwingt das Sammeln aller ererbten Attribute innerhalb der zu instantiierenden Instanz. Die ererbten Attribute werden den lokal definierten Attribute vorangestellt. Für alle Instanzen, die kein Leaf-Node des Suptyp/Supertyp-Graphen darstellen, wird das externe Mapping angewandt: Alle Entitäten, die zur Bildung der Instanz beitragen werden selbst instantiiert und in eine Liste von Instanzen eingefügt.
  Nach der Konformitätsklasse 2 müssen alle Instanzen nach dem externen Mapping geschrieben werden.
- Langnamen oder Kurznamen als Entitybezeichner
  Die oben eingeführte Abbildungstabelle zwischen den Langnamen und Kurznamen der in einem Schema definierten Entitäten ermöglicht es, für Instanzen von Entitäten innerhalb einer STEP-Datei Kurznamen der instantiierten Entitäten zu verwenden. Nach /43/ ist dies sogar für konforme Dateien gefordert. Alternativ sind jedoch auch die innerhalb des Schemas verwendeten Entitybezeichner möglich. Damit sind zwei weitere Formate definiert.
- Das in /43/ eingeführte Scope-Konstrukt stellt ein durch das Modellschema nicht gerechtfertigtes Modellierungskonstrukt bereit. Zwei Datensätze, die bis auf unterschiedliche Scope-Konstrukte identische Instanzenmengen enthalten sind hinsichtlich der Modellbildung nicht zu unterscheiden. Innerhalb der Austauschdatei existierende Scopes sind durch kein Modellkonstrukt rekonstruierbar. Damit stellt ein Datensatz, aus dem jegliche Scope-Konstrukte entfernt wurde, ebenfalls nur eine andere Schreibweise eines Datensatzes dar.

Mit dem STEP-Dateiformatierer wurde ein Backend zu dem STEP-Dateiparser realisiert, der die STEP-Working-Form orthogonal zu den Formatprinzipien formatieren kann. Durch Angabe von Optionen können aus Datensätzen mit Langnamen solche mit Kurznamen und umgekehrt, aus Datensätzen nach Konformitätsklasse 1 solche nach Kon-

formitätsklasse 2 und umgekehrt und aus Datensätzen, die Scope-Konstrukte enthalten solche ohne Scope-Konstrukte erzeugt werden. Eine Erzeugung von Scope-Konstrukten hingegen ist sinnlos.

3.3.4 Abbildung von EXPRESS-Modellen in eine Softwareentwicklungsumgebung

Die Basis einer effizienten Entwicklungsumgebung muß durch eine Grundfunktionalität im Rahmen der EXPRESS-Modellierung bereitgestellt werden. Graphisch orientierte Arbeitstechniken erleichtern das Verständnis der Struktur eines Modells und die Zusammenhänge der Modellelemente.

Die in Kapitel 3.3 definierten Anforderungen stellen einerseits allgemeine Anforderungen an Entwicklungswerkzeuge dar, die durch anwendungsflexible Systeme bereits großteils unterstützt werden können (z.B. Anbindung an Programmiersprachen, Datenbankanbindung, Benutzeroberfläche) andererseits eine anwendungsorientierte Erweiterbarkeit erforderlich machen. Für die Anwendbarkeit einer generischen Entwicklungsumgebung spielt dabei eine Rolle, inwieweit EXPRESS/STEP-Paradigmen innerhalb des Basissystems bereits unterstützt werden, bzw. welche Einschränkungen dieser Konzepte zu erwarten sind.

Die Entwicklungsumgebung Kappa von Intellicorp /193/ stellt eine anwendungsflexible Umgebung zur Entwicklung von DV-Systemen dar. Sie besteht aus den in Abb. 17 dargestellten Modulen.

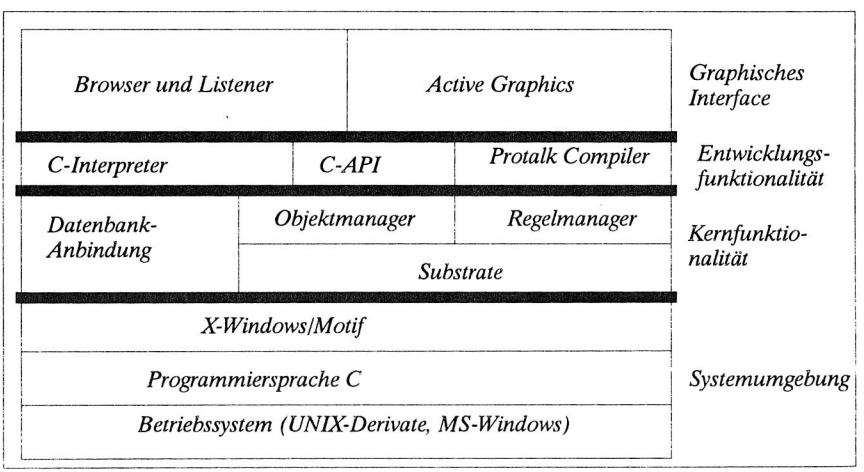


Abb. 17 Kappa Systemarchitektur (nach /193/)

Kappa bietet ein C-basierte Programmierumgebung, die u.a. eigene Datentypen (siehe Abb. 18), Binden zur Laufzeit, Speicherverwaltung und programmierbare Fehlerbehandlung unterstützt. Einer der für die realisierte Umgebung wichtigsten Baustein stellt das Kappa Objektsystem dar. Das Kappa Objektsystem organisiert Konstrukte des Problemraums in Objekthierarchien. Diese hierarchische Strukturen unterstützen mehrfache und tiefe Vererbung. Daten eines Anwendungsobjektes lassen sich in sogenannten Slots und Facets beschreiben. Die Kommunikation der Objekte erfolgt über den Austausch von Nachrichten (Message Passing). Das Verhalten von Objekten läßt sich mit Hilfe von Monitoren lokal steuern.

Die Grundfrage jeder Abbildung auf eine Laufzeitumgebung ist das Konzept der Abbildung von EXPRESS-Konstrukten auf Konstrukte der Zielumgebung. Die prinzipielle Frage hierbei ist, ob die Überführung von EXPRESS-Konstrukten in jeweils naheliegende Konstrukte der Zielumgebung geschehen soll, oder ob mit Hilfe des Zielsystems eigene, den EXPRESS-Konstrukten möglichst identische neue Konstrukte geschaffen werden sollen, um eine semantiktreue Abbildung zu gewährleisten. Die Entscheidungskriterien für diese Alternativen sind im wesentlichen vom Einsatzzweck der Realisierung abhängig:

Handelt es sich um eine Abbildung, die eine Implementierung eines Anwendungssystems, das STEP-konform sein soll, so gelten im wesentlichen Kriterien, die eine akzeptable Performanz des Systems präferieren. Hierbei sind strukturelle Abbildungen vorzuziehen, auch um den Preis gewisser konzeptverletzender Abbildungsprinzipien. Als Beispiel hierfür kann die Definition eines EXPRESS-Typs herangezogen werden:

```
TYPE positive = INTEGER
WHERE SELF > 0;
END_TYPE;
```

Bei semantischer Übertragung dieser Typdefinition in eine Implementierung die beispielsweise in C oder C++ vorgenommen werden soll könnte diese Typdefinition durch:

```
typedef unsigned int positive;
```

abgebildet werden. Dies würde jedoch ein Verständnis der Semantik der Definition voraussetzen. Mit der Forderung einer maschinellen Übersetzung eines EXPRESS-Schemas würde daher eine semantische Übertragung gefordert werden. Aus der Compilertechnologie /135/ ist jedoch zu entnehmen, daß solche Übersetzungen für komplexe Sachverhalte im allgemeinen nicht möglich sind.

Darüberhinaus ist bei einer derartigen Abbildung bereits ein Informationsverlust aufgetreten. Da EXPRESS keine Implementierungsorientierung besitzt, ist der Wertebereich der Zahlentypen nicht begrenzt. Eine Abbildung auf eine implementierungsspezifischen Typ hat daher einen Informationsverlust zur Folge.

Mit der in Kapitel 3.3 definierten Anforderungen wurden für die Abbildung von EXPRESS-Schemata auf Kappa folgenden Präferenzen definiert:

- maschinelle Übersetzbarkeit der EXPRESS-Schemata
- Abbildung auf strukturelle Ebene des Zielsystems
- Abbildung auf systeminterne Konstrukte und Typen und
- Beschreibung der EXPRESS-Schemata in Metainformationsmodellen.

Das letztgenannte Kriterium soll die Möglichkeit bieten, unter bestimmten Voraussetzungen ein EXPRESS-Modell vollständig, widerspruchsfrei und eindeutig beschreiben zu können. Die in /196/ für EXPRESS entwickelten Metamodelle bieten hierfür Ansätze. Interessant für eine maschinelle Abbildung in ein Laufzeitsystem ist die Definition dieser Metamodelle in EXPRESS selbst. Die Anwendbarkeit der Metamodelldefinition wie sie in /44/ für die Definition des SDAI spezifiziert wird, wird bei der Modellabbildung nach Kappa genutzt. Dies hat einerseits den Erhalt der Modellsemantik während der Abbildung zum Ziel und andererseits die direkte Nutzbarkeit der Datenbasis als STEP-konforme Produktdatenbasis.

Für die Abbildung von EXPRESS-Modellen muß untersucht werden, welches EXPRESS-Konstrukt sich auf welches Kappa-Konstrukt abbilden läßt. Aus der in Abb. 18 ersichtlichen Typhierarchie und der für sie dokumentierten Semantik läßt sich entnehmen, daß wesentliche Konzepte von EXPRESS direkt in Kappa abbildbar sind. Die einfachen Typen von EXPRESS: INTEGER, REAL, NUMBER und STRING sowie die Aggregationstypen ARRAY und LIST finden ihre direkten Abbildungen in den Kappa Typen: *PrkFixnum*, *PrkFloat*, *PrkNumber*, respektive *PrkArray* und *PrkList (siehe Abb. 18)*.

Für die Abbildung der EXPRESS Typen LOGICAL und BOOLEAN wurde die Abbildung in *PrkString* vorgenommen; Wertvergleiche von Enumerationswerten müssen dann jedoch durch Rückgriff auf die Typdefinition im *Data Dictionary* ausgeführt werden. Für die Abbildung der Typen BAG und SET wurde *PrkList* gewählt. Die auf diesen EXPRESS Typen definierte Semantik wird durch den Kappa Typ *PrkList* unterstützt. Die in EXPRESS definierbare Unikatseigenschaft von Werten in ARRAY und LIST läßt sich hingegen in Kappa nur funktional erreichen. Eine Spezifikation der Basistypen von Kappa Aggregaten (PrkCollection) ist nicht möglich; letztendlich kann in Kappa ein *PrkCollection* jeden beliebigen Typ, gemischt auch innerhalb einer *PrkCollection* aufnehmen.

Die Abbildung des durch die in einem EXPRESS Modell definierten Subtyp/Supertyp-Graphen stellt den wesentlichen Anteil an einer sinnvollen Schemaabbildung in Kappa dar. Dabei muß die Semantik der Objekthierarchie in Kappa und deren Funktionalität mit der Semantik des durch EXPRESS Subtyp/Supertyp-Graphen näher verglichen werden.

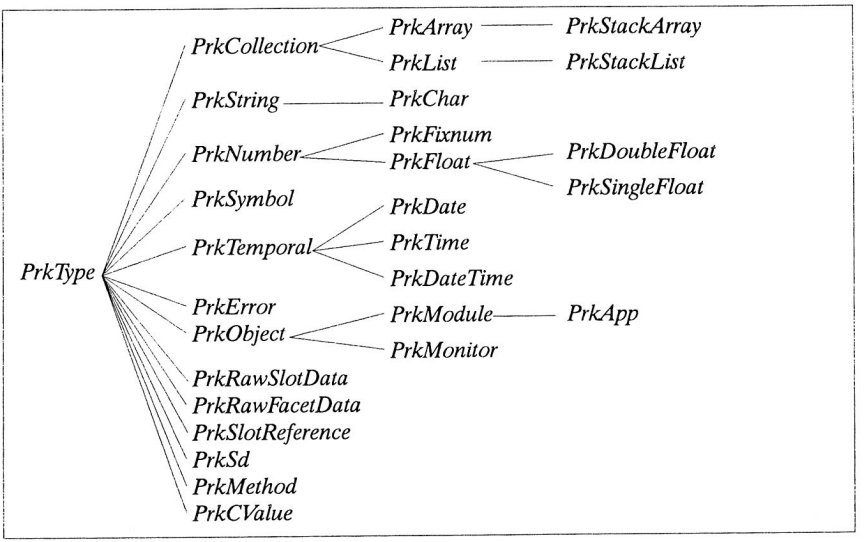


Abb. 18 Typhierarchie in Kappa (nach /193/)

Entitäten in EXPRESS tragen Attribute von beliebigen, in EXPRESS formulierbaren Typen. Objekte in Kappa besitzen Slots, in denen Werte beliebiger, in Kappa definierbarer Typen gespeichert werden können. Kappa unterscheidet nur drei Typen von Slots:

- *Single Value Slots*: Der Wert eines Single Value Slots kann jeden beliebigen Kappa Typ aufnehmen.
- *Multi Value Slots*: In einen Multi Value Slot können beliebige Mengen von Werten beliebiger Kappa Typen aufgenommen werden. Er entspricht einer *PrkList*.
- *Method Slots* dienen zur Aufnahme von Methoden in ein Kappa Objekt.

Die Einschränkung eines Slots auf einen Wert eines bestimmten Typs ist in Kappa strukturell nicht formulierbar. Hingegen könnte ein Facet (Fasette) eines Slots den gewünschten Typ eines Wertes für einen Slot aufnehmen. Sollte ein Wert eines Slots gesetzt oder geändert werden, müßte funktional überprüft werden, ob der Wert dem Wertebereich des gewünschten Typs entspricht.

EXPRESS Entitäten lassen sich in zyklenfreien Hierarchien anordnen, wobei tiefe und mehrfache Vererbung definiert ist. Dies ist in Kappa durch die Objekthierarchie direkt nachbildbar. Kappa Objekte sind entweder Klassen oder Instanzen. Kappa Klassen können als Klassen im allgemeinen objektorientierten Verständnis /89/ aufgefaßt werden, allerdings können Kappa Klassen im Gegensatz zu Klassen wie etwa in C++ selbst Informationsträger und Handlungsobjekte sein. Kappa Instanzen sind spezielle Objekte, die an den Enden einer Objekthierarchie stehen. Sie können nicht 'Superobjekte' weiterer Objekte sein und stellen somit die Blätter der Objekthierarchie dar. Kappa

Objekte unterstützen tiefe und mehrfache Vererbung. Die Vererbung als solches ermöglicht nicht nur die Propagierung von Slots (Attributen), sondern darüberhinaus auch die Vererbung von Werten (Attributwerten) entlang des Vererbungspfades.

Da auch Kappa Instanzen mehrfache Vererbung unterstützen, ist das in der EXPRESS-Welt problematische Erzeugen von Instanzen mehrerer Entitäten (komplexe Entityinstanzen) unproblematisch. Die Einschränkung der Instanzenbildung auf die Menge ausgewählter Klassen ist hingegen in Kappa strukturell nicht formulierbar.

Für die in EXPRESS definierbaren abgeleiteten Attribute läßt sich in Kappa die Berechnung eines Wertes über sogenannte Monitore steuern. Monitore lassen sich Slots zuordnen und lösen eine Aktivität aus. Die Aktivierung der Aktion kann vor oder nach einem Änderungszugriff, oder vor einem Lesezugriff ausgeführt werden. Da in EXPRESS ein Änderung eines abgeleiteten Attributs nicht definiert ist, ist die Definition eines Lesezugiffmonitors mit der Berechnungsvorschrift auf den Slot eines abgeleiteten (DERIVED) Attributs die adäquate Abbildung eines abgeleiteten EXPRESS Attributes. Inverse (INVERSE) Attribute in EXPRESS sind in Kappa durch die Referenz(en) der bezogenen Attribute darstellbar. Eine Unterstützung der Konsistenzüberprüfung der Kardinalität der inversen Attribute existiert hierfür allerdings nicht. Dies ist durch funktionale Anteile der Abbildung sicherzustellen.

EXPRESS-Konstrukt	*Kappa-Konstrukt*
SCHEMA	*PrkModule*
INTEGER	*PrkFixnum*
REAL	*PrkDoubleFloat*
NUMBER	*PrkNumber*
STRING/BINARY	*PrkString*
LOGICAL/BOOLEAN	*PrkFixnum*
ARRAY	*PrkArray*
LIST/BAG/SET	*PrkList*
ENTITY	*PrkObjekt*
TYPE	*(Basistyp des benannten Typs)*
ENUMERATION	*Aufzählwerte als PrkString*
SELECT	*PrkType*
DERIVE	*Berechnung über PrkMonitor/PrkMethod*
RULE/WHERE	*Berechnung über PrkMethod*
Supertyp/Subtyp-Hierarchie	*Kappa-Objekthierachie*

Tab. 3 Abbildung von EXPRESS-Konstrukten auf Kappa Konstrukte

Optionalität (OPTIONAL) von Elementen von aggregatwertigen Typen bzw. Werten von Attributen kann in Kappa prinzipiell dargestellt werden, da ein Kappa-Slot nicht notwendigerweise mit einem Wert belegt sein muß. Andererseits kann die Verbindlichkeit eines Wertes in Kappa strukturell nicht formuliert werden.

Für die Abbildung von EXPRESS SCHEMA in Kappa bieten sich die Konstrukte *Module* und *Application* an. Kappa *Applications* (*PrkApp*) stellen im Sinne der DV-Entwicklung abgeschlossene Einheiten dar, die für sich genommen, lauffähige Anwendungen bilden. Kappa *Module* (*PrkModule*) bestehen darin als Konzepte, die eine nicht notwendigerweise abgeschlossene Untereinheit bilden. Wesentliche Eigenschaft der *Applications* und *Modules* sind der abgeschlossene Namensraum, der eine eindeutige Zuordnung eines Symbols zu einem Modul bzw. einer Applikation ermöglicht. Dies erlaubt die Abbildung von EXPRESS SCHEMAs auf Module. Weiterhin lassen sich durch die Hierarchie von *Applications* und *Modules* die Struktur von Schemas abbilden, die andere Schemas referenzieren.

Die Abbildung von EXPRESS Auswahltypen (SELECT) stellt für viele Zielsysteme ein Problem dar, da die Basistypen des Auswahltyps sowohl Entitäten als auch skalare Werte oder Aggregate sein können. Für Kappa hingegen stellt sich diese Abbildung höchst einfach dar, da der hierzu benutzte Kappa Typ *PrkType* eine Generalisierung jedes Kappa Typs darstellt (Abb. 18).

Benannte Typen können im Kappa Typsystem nicht direkt abgebildet werden; hierfür wurde die indirekte Abbildung über das Data Dictionary gewählt. Die Verwendung von benannten Typen wurde auf die Abbildung des Basistyps des benannten Typs zurückgeführt.

Für die maschinelle Übersetzung eines EXPRESS-Models in Kappa wurde eine Backend des EXPRESS-Parsers Expparse (Kapitel 3.3.2) realisiert, der den Parse-Tree (Working Form) traversiert und die entsprechenden Konstrukte in Kappa erzeugt. Vor dem Hintergrund einer semantiktreuen Abbildung von EXPRESS-Modellen in Kappa werden sowohl die EXPRESS-Metamodelle die aus der SDAI-Spezifikation /44/ hervorgehen sowie das 'EXPRESS Semantic Meta Model' (ESMM) /196/ benutzt, um Konstrukte abzubilden, die nicht direkt Eingang in die Objektstruktur von Kappa finden.

Hierfür wurde ein zweistufiger Abbildungsvorgang benutzt: In einem ersten Übertragungsvorgang wird das Modell des SDAI Data Dictionary abgebildet. Hierbei werden zunächst nur die Supertyp/Subtyp-Graphen des SDAI Data Dictionary als Objektstrukturen in Kappa abgebildet (siehe Abb. 19).

In einem zweiten Schritt kann dann jedes korrekte EXPRESS-Modell in Kappa abgebildet werden. Die einzelnen EXPRESS-Konstrukte des abzubildenden Schemas werden nach den in Tab. 3 dargestellten Mechanismen transformiert. In Kapitel 3.3.6 wird gezeigt, wie in STEP-Dateien definierte Datensätze in die Kappa Objektbasis überführt werden.

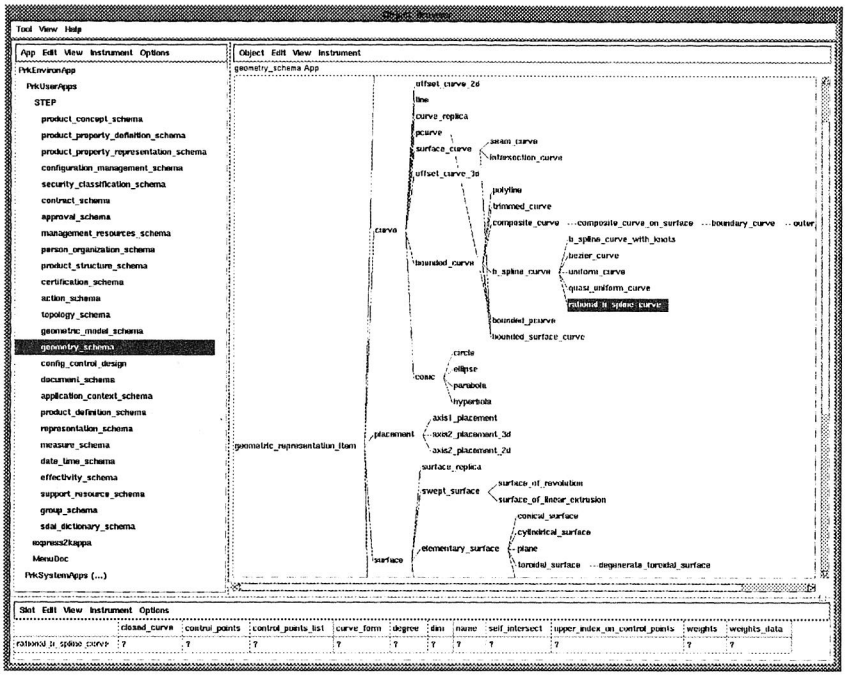


Abb. 19 Abbildung von EXPRESS-Modellen in Kappa

Abb. 19 zeigt einen Ausschnitt des nach Kappa transformierten EXPRESS-Modells des AP203 /61/. In der Abbildung links sind die erzeugten Module zu sehen, die den EXPRESS-Schemata entsprechen. Zu beachten ist das SDAI Data Dictionary (*sdai_dictionary_schema*) zur Beschreibung der im EXPRESS-Modell definierten Typen. Im rechten Bildabschnitt ist ein Ausschnitt des Supertyp-Subtyp Graphen des Schemas 'geometry_schema' dargestellt, der in Kappa durch die Kappa Objekthierarchie repräsentiert wird. Im unteren Bildabschnitt ist die Kappa Repräsentation der Entität 'rational_b_spline_curve' mit ihren Attributen (in Kappa 'Slots') dargestellt. Für dieses Kappa Objekt sind sowohl die lokal definierten, als auch die ererbten Attribute verfügbar.

3.3.5 Instantiierung des Datenmodells

Mit der Abbildung eines EXPRESS-Schemas nach Kappa ist es gelungen, ein Datenmodellschema in Kappa abzubilden. Die Erzeugung und Verwaltung von Daten, die dem gebildeten Schema entsprechen, kann durch unterschiedliche Mechanismen realisiert werden:

- Durch Einbringen und Ausleiten der Information in STEP-Austauschdateien über Pre- und Postprozessoren;
- Anwendungen, die die Kappa Objekte über das Kappa API manipulieren;
- Anwendungen, die auf die Kappa Objekte über ein SDAI zugreifen oder
- durch interaktives Manipulieren der Kappa Objekte über die Kappa Entwicklungsumgebung.

Der Fall der Einbringung und Ausleitung von Daten mit Hilfe von STEP-Austauschdateien wird weiter in Kapitel 3.3.6 beschrieben.

Ein direktes Manipulieren der Information über das API von Kappa stellt die einfachste und effektivste Methode dar, eine Anwendung zu realisieren, die mit STEP-kompatiblen Daten arbeitet. Die Definition einer STEP-kompatiblen Anwendung ist jedoch nicht von der Strukturierung der Daten innerhalb der Anwendung, sondern von der Bereitstellung oder Nutzung der definierten bzw. genormten Schnittstellen wie sie durch die STEP-Implementierungsmethoden der Austauschdateien bzw. des SDAI gegeben sind, abhängig. Daher ist eine STEP-orientierte Anwendungsentwicklung auf der Kappa API nicht sinnvoll und wird daher nicht weiter betrachtet.

Die Nutzung von STEP-kompatiblen Daten über einem SDAI setzt die Implementierung eines SDAI über der zugrundeliegenden Basis voraus. Die Realisierung und Einbindung eines SDAIs in Kappa wird in Kapitel 3.3.7 weiter verfolgt.

Die zuletzt angesprochene Möglichkeit der Erzeugung von Instanzen aus der Entwicklungsoberfläche von Kappa heraus, muß sich natürlich der oben erwähnten Kritik unterordnen. Als einfache Möglichkeit der Instantiierung und Wertzuweisung bietet sie STEP-Entwicklern jedoch einen einfachen Zugriff auf die durch EXPRESS-Modelle definierten Daten.

Die Erzeugung von Instanzen eines in Kappa abgebildeten ENTITY läßt sich zunächst als die Instantiierung eines Kappa Objektes verstehen. Durch die Semantik von EXPRESS ist festgelegt, daß sich zu einem ENTITY eine Instanz erzeugen läßt, solange es nicht als abstrakt (ABSTRACT) gekennzeichnet ist. Dies entspricht dem Erzeugen einer Instanz einer Kappa Klasse. Die Nichtinstantiierbarkeit einer Klasse läßt sich in Kappa nicht strukturell formulieren. Hierfür wäre die Definition einer Funktion nötig, die die Instantiierung von Objekten überwacht.

EXPRESS erlaubt durch die Formulierung von Constraints in SUPERTYPE Klauseln (ONEOF, AND, ANDOR) die Definition von Instanzen sogenannter komplexer Entitäten. Diese werden durch in /41/ definierten Regeln aus den durch SUPERTYPE/SUBTYPE verbundenen Entitäten gebildet. Durch die Auswertung der Regel wird bestimmt, mit welchen Entitäten als mehrfache Superklassen Instanzen gebildet werden können. Da in Kappa zunächst Instanzen mit Vorgängern beliebigen Klassen gebildet werden können und keine strukturelle Einschränkung dieser Hierarchisierung möglich ist, muß die Abbildung der SUPERTYPE/SUBTYPE Constraints funktional vorgenommen werden.

3.3.6 Abbildung von STEP-Datensätzen in die Entwicklungsumgebung

Wie in Kapitel 3.3.5 verdeutlicht, ist eine der Möglichkeiten, Informationen in eine STEP-Datenbasis einzubringen, das Lesen von Austauschdateien. Mit Hilfe des Austauschdateiparsers können in das vorgestellte System Daten, die als Instanzen von EXPRESS-Entitäten beschrieben wurden, eingebracht werden.

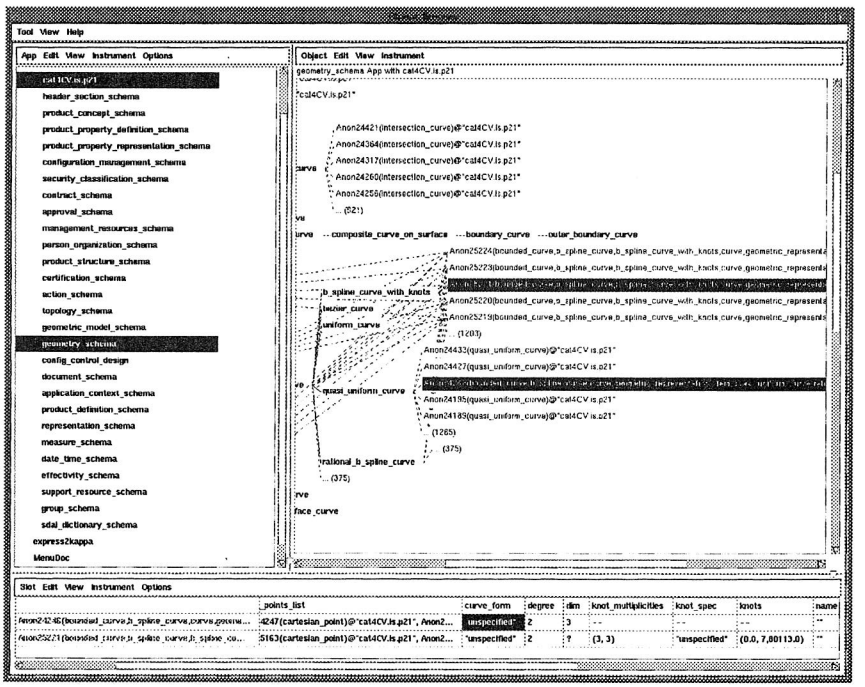


Abb. 20 Instanzen von EXPRESS-Entitäten als Instanzen von Kappa-Objekten

Die in den Austauschdateien enthaltenen Instanzen von EXPRESS Entitäten werden in Kappa auf Instanzen von Kappa-Klassen abgebildet. D.h. die Instanz

```
#3  = COORDINATE_POINT ('endpoint1',(1.23, 4,56,7.89));
```

der EXPRESS-Entität *coordinate_point* wird zu einer Instanz der Kappa Klasse *coordinate_point*. Der Instanzenidentifikator #3 innerhalb des Austauschdatei hat nur innerhalb des Kontextes der Übertragungsdatei Gültigkeit. Innerhalb einer Datenbasis bzw. für ein Anwendungssystem, das mittels SDAI auf diese Datenbasis zugreift, ist dieser Identifikator nicht nötig, bzw. sichtbar. Für die Benennung einer Instanz in der Datenbasis ist der Erhalt des Identifikators der Übertragungsdatei nicht nötig. Für die vorgenommene Einbringung der Instanzen wurde daher der folgende Abbildungsmechanismus gewählt: Für jede erzeugte Instanz wird ein von Kappa erzeugter eindeutiger Identifika-

tor vergeben. Dieser Identifikator wird aus einem Surrogat aus dem Klassennamen der Superklasse der Instanz und einer hochlaufenden natürlichen Zahl gebildet.

In Abb. 20 ist im rechten Bildabschnitt eine Menge von abgebildeten Instanzen zu erkennen. Dabei wurden zwei komplexe Instanzen ausgewählt, deren Attributwerte im unteren Bildabschnitt sichtbar sind.

Die Zuordnung der Instanzen zu einer Datenverwaltungseinheit, die durch das SDAI *Repository* bestimmt wird ist durch die Einordnung der Instanzen in ein eigenes Kappa Modul möglich.
Durch Kappa können einzelne Module auf Sekundärspeichermedien ausgelagert und ebenso wieder geladen werden. Hierdurch ist eine einfache Datenverwaltungsfunktionalität realisiert.

3.3.7 Generierung der schemaabhängigen C++-Klassen des SDAI

Wie in Kapitel 3.3.5 erläutert ist die flexible Nutzbarkeit von STEP-kompatiblen Anwendung an die Kopplung über die Schnittstelle SDAI gebunden. Als ein STEP-konformes Basissystem muß eine STEP-Datenbasis über ein SDAI einer gewählten Implementierungssprache verfügen. Die nach den Implementierungssprachen unterschiedenen SDAI-Spezifikationen sind für ein gewähltes Basissystem nach folgenden Kriterien zu unterscheiden:

- Type Mismatch zwischen Implementierungssprache und Basissystem
- Type Mismatch zwischen SDAI-Implementierung und Anwendungssystem
- Performanz der SDAI-Spezifikation

Für die beiden ersten Punkte sind die vorhandenen Systeme die als Datenverwaltungssystem herangezogen werden, sowie die Systeme, die als Anwendungssysteme zu erwarten sind, zu untersuchen. Für den dritten Punkt kann eine davon unabhängige Aussage getroffen werden: Die für das C++-Binding definierte early-Binding muß gegenüber den Alternativen einen prinzipielle Performanzvorteil besitzen. Die soll an einem Beispiel erläutert werden:

Gegeben sei eine Entity-Definition mit:

```
ENTIY point;
x       :       REAL;
y       :       REAL;
z       :       OPTIONAL REAL;
END_ENTITY;
```

so wird für das C++ early-Binding zum setzen des Attributes x einer Instanz (hier mit dem Bezeichner Point_inst definiert) der Entität point auf einen Wert folgende Funktion

definiert:
class Point;
{...
SetValue(Point_inst, x, SdaiReal);
...
}

Daraus wird deutlich, daß zum Setzen, Verändern und Lesen von Werten von Instanzen einer STEP-Produktdatenbasis unter Nutzung des early-Bindings, das für C++ definiert ist, ein Funktionsaufruf weniger nötig ist, als bei den anderen bisher definierten Sprachanbindungen.

Darüberhinaus ist über die in C++ sprachseitig definierte Typüberprüfung zur Kompilierzeit zur Realisierung typsicherer Anwendungen ein Beitrag zur modernen Softwareentwicklung.

Das early binding des C++ language binding definiert für jede Entität des Anwendungsschemas eine eigene C++-Klasse. Jede dieser Klassen ist eine Unterklasse von all den Klassen, die aus Entitäten hervorgehen, die innerhalb der EXPRESS-Schemas im Supertype/Subtype-Graphen oberhalb der zu erzeugenden Entität stehen. Für jede Klasse werden dann für jedes Attribut eigene Put-, Get-, Test- und Unset- (Member-) Funktionen definiert. Für die folgende EXPRESS ENTITY-Definition:

```
ENTITY
END_ENTITY;
```

ergibt sich damit folgende C++ Klassendefinition:
...
Automatisch werden für jede derart definierte Klasse Standardkonstruktoren, Copy-Konstruktoren und Destruktoren definiert. Im Rahmen der eigenen Implementierung wurde festgestellt, daß jedoch kein Konstruktor definiert wurde, der durch Übergabe der Attribute einer Entität eine Instanz der jeweiligen Klasse bildet. Dies sollte bisher nur durch Erzeugen der Klasse mit Hilfe des Standard-Konstruktors und nachträglichem Setzen der Attributwerte möglich sein. Der hierdurch unnötige Performanznachteil kann durch die Definition eines Konstruktors, der Werte für alle Attribute aufnimmt, geheilt werden. Eine notwendige Voraussetzung zur Definition und Anwendung dieses Konstruktors ist die Definition aller, auch der ererbten Attribute in das Data Dictionary. Dies wurde zur Aufnahme in die Normdefinition vorgeschlagen.

Das Prinzip der C++ SDAI-Spezifikation realisiert den Ansatz, alle Konzepte, die keine direkten Äquivalenzen in C++-Konstrukten finden, durch C++-Klassendefinitionen zu abstrahieren. Direkte Abbildungen werden dabei nur für die einfachen Typen INTEGER (long int) und REAL (double) vorgenommen.

Die Generierung des SDAI läßt sich prinzipiell aus zwei Umgebungen heraus realisieren:

- Direkte Generierung aus dem EXPRESS-Schema.
- Generierung aus dem für ein oder mehrere Schemata gefüllten Data Dictionary des Basissystems des SDAI-Trägers heraus.

Für das erste Prinzip spricht die direkte Generierung ohne den Umweg über ein Zwischensystem nehmen zu müssen, für die zweite Alternative die Tatsache, daß bei der Generierung aus dem SDAI-Trägersystem heraus, die Eigenschaften des SDAI-Trägers bekannt sind und diese während der SDAI-Generierung berücksichtigt werden können. Stellt das Basissystem darüberhinaus Software-Engineering-Funktionalität zur Verfügung, so wird der Nachteil einer zweifachen Abbildung zumindest relativiert. In der hier vorgestellten Umgebung ist die Generierung des SDAIs aus der Entwicklungsumgebung heraus ein Teil deren Funktionalität.

In Abb. 21 wird das Verfahren der realisierten Generierung der SDAI-Implementierung verdeutlicht. Mit Hilfe eines Kappa Anwendungsprogrammes *SDAIgen* wird aus dem in Kappa abgebildeten SDAI Data Dictionary, das einerseits die Beschreibung des SDAI Data Dictionary selbst und andererseits die Beschreibung eines Anwendungsmodells enthält, generiert. Anteile der SDAI-Spezifikation, die einer automatisierten Code-Generierung nicht zugänglich sind, müssen darüberhinaus manuell codiert werden. Automatisiertes compilieren des erzeugten C++-Quelltextes schließt sich daran an.

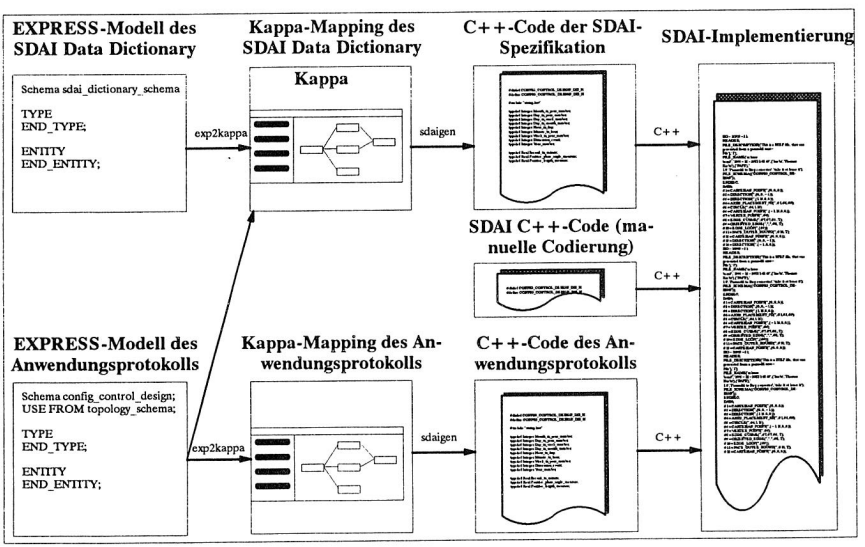


Abb. 21 Prinzip der automatisierten SDAI-Generierung

Die Spezifikation des C++-SDAI beschreibt zunächst nur eine Kompatibilität auf der deklarativen Ebene der Implementierung. Die Realisierungen C++-SDAIs können daher in gewissem Umfange unterschiedlich sein. Die Kompatibilität wird daher nur durch

gleichlautende aber unterschiedlich implementierte C++-Konstrukte auf Quelltextebene möglich sein. Linkkompatibilität wird dabei nicht angestrebt.

Das Basissystem, für das eine SDAI-Anbindung realisiert werden soll, ist in dem vorliegenden Fall das Entwicklungssystem selbst. Die Funktionalität von Kappa, die Kappa internen Daten persistent zu verwalten, stellt dabei eine der wesentlichen Anforderungen an ein SDAI-Basissystem sicher.

Während die Deklarationen einer SDAI-Realisierung zwar Implementierungsdetails vorwegnehmen, sind sie bezüglich des Basissystems abstrakt.
Der Implementierungcode hingegen ist immer auf das zugrundeliegende Basissystem bezogen. Im vorliegenden Fall stellen die realisierten Methoden der Erzeugung (Konstruktor) und Vernichtung (Destruktor) von Entityinstanzen sowie das Setzen und Abfragen von Attributwerten von Entityinstanzen eine Schale um die Funktionen zur Erzeugung und Vernichtung von Kappa Objekten sowie um das Setzen und Erfragen von Kappa Slotwerten dar.

Mit der realisierten SDAI-Zugriffsschicht werden über eine SDAI-Schale die Objektverwaltungsfunktionen von Kappa von einem Anwendungssystem benutzt, ohne Kenntnis über die Existenz dieses Basissystems zu haben. Obwohl diese Abstraktion von dem Basissystem Sinn einer Zugriffsschicht wie SDAI ist, bietet die realisierte Konstruktion die Möglichkeit, mit Hilfe der Kappa Entwicklungsumgebung abgelegte STEP-Datensätze interaktiv zu manipulieren, und für Rapid-Prototype-Entwicklungen zu nutzen.

3.3.8 Editor für EXPRESS Quelltexte

Die Modellbildung eines EXPRESS-Schemas wird vorzugsweise in grafischer Form mit Hilfe der grafischen EXPRESS-Untermenge EXPRESS-G vorgenommen (siehe Kapitel 2.3.2). Die nicht in EXPRESS-G darstellbaren algorithmischen Anteile eines Modells müssen darüberhinaus jedoch weiterhin in der lexikalischen Form in ein Modell eingebracht werden. Hierfür sind Texteditoren sinnvoll einzusetzen.

Durch Fähigkeiten von Editoren können folgende Funktionen in Editoren angeboten werden:

- Verzweigen zu (von EXPRESS-Compilern) identifizierten fehlerhaften Textpositionen in EXPRESS-Quelltexten.
- Hervorgehobene (farbig, Schriftart) Darstellung von Schlüsselwörtern, Sonderzeichen und internen Funktionsnamen.
- Kontextsensitives Editieren eines EXPRESS-Quelltextes.
- Verzweigen zu referenzierten Konstrukten.

Mit Hilfe des Editors Emacs /183/ wurde ein EXPRESS-Editor realisiert, der die oben identifizierten Mechanismen unterstützt:

Durch die von *Expparse* erzeugten Fehlermeldungen beim Parsen eines EXPRESS-Quelltextes ist der Editor in der Lage die betroffene Datei und die Zeile des Ortes der Fehlers zu lokalisieren und die aktuelle Pufferposition auf den betroffenen Bereich zu setzen.

Die kontextsensitive Darstellung eines Quelltextes wird mit Hilfe von Emacs-Lisp-Paketen unterstützt. Das Packet zur farblichen Hervorhebung von Zeichenketten (hilit19) benötigt Listen von Zeichensequenzen, die unterschiedlichen Klassen von Zeichenketten zugeordnet und gemeinsame Eigenschaften besitzen. Es wurden für EXPRESS folgende Klassen mit den Zuordnungen geschaffen:

- Schlüsselwörter (ALIAS, CASE, CONSTANT, CONTEXT, ENTITY, FUNCTION usw.)
- Typbezeichner (AGGREGATE, ARRAY, BAG, LIST, SET, SELECT, BINARY usw.)
- Operatoren mit textueller Darstellung (AND, ANDOR, DIV, IN, LIKE, MOD, NOT usw,)
- Konstanten (CONST_E, PI, FALSE, TRUE, UNKNOWN, ?, SELF)
- Interne Funktionen (ABS, ACOS, ASIN, ATAN, BLENGTH, usw.)
- Prozeduren (INSERT, REMOVE)
- Interfacedefinitionen (USE, REFERENCE)

Die Ausprägungen der Klassen werden z.B. mit identischen Farben und Schriftarten dargestellt[11].

Das Schreiben korrekter EXPRESS-Quelltexte kann darüberhinaus durch kontextsensitives Editierunterstützung erleichtert werden. Unter die Hilfefunktionen fallen u.a. Identifikation korrespondierender Klammerpaare und konstruktgerechtes Einrücken in Folgezeilen.

Mit Hilfe sogenannter Tags-Dateien kann der Editor Verzeigerungen auf Bezeichner der Quelltextes verwalten. Durch Selektion eines Bezeichners verzweigt der Editor dann zur Stelle der Definition des Bezeichners, wobei die Darstellung auch in einem zweiten Puffer des Editors erfolgen kann.
Das Erzeugen der TAGS-Dateien kann mit Hilfe des UNIX-Dienstprogrammes *etags* erfolgen, das für verschiedene Sprachen (C, C++, FORTRAN usw.) direkte Unterstützung beinhaltet, aber auch durch Angabe von regulären Ausdrücken zur Generierung von TAGS-Dateien für EXPRESS-Quelltexte verwendet werden kann.

---

11. Dies setzt natürlich Fähigkeiten des Terminals zur Darstellung farbiger oder unterschiedlicher Zeichensätze voraus.

4 Datenbankimplementierungen von STEP-Datenbasen

Über die Definition eines Datenaustauschformates hinaus erlaubt die STEP-Methodik die Nutzung von Produktdatenbasen zur zentralen oder verteilten Verwaltung von Produktdaten. Wesentliches Hilfsmittel zur Verwaltung von Produktdaten können Datenbankmanagementsysteme (DBMS) sein /194/. Die weitreichende Akzeptanz von DBMS läßt deren Einsatz als unternehmensübergreifende CIM-Bausteine rechtfertigen. Die eingeschränkte Anwendung von Datenbanken als Informationszentren für die technische Produktdatenbeschreibung war bisher vor allem durch unterschiedliche Modellierungsprinzipien für Anwendungssysteme der kommerziellen und der technischen Datenverarbeitung begründet.

Im Folgenden soll die Nutzung von Datenbankmanagementsystemen als STEP-Produktdatenbanken dargestellt werden. Die heute relevanten Modelle der DBMS werden vorgestellt und hinsichtlich ihrer Eignung für STEP-Produktdatenbanken untersucht. Als Realisierungsumgebung wird das Erweitert Relationale Datenbankmanagementsystem Postgres vorgestellt und die Implementierung einer STEP-Produktdatenbank erläutert.

4.1 Datenbankprinzipien im Vergleich

Dem Einsatz eines DBMS in einem gegebenen Aufgabenbereich muß ein Abgleich der vom DBMS angebotenen Funktionalität mit den gestellten Anforderungen vorausgehen. Dies muß prinzipiell für jedes relevante System explizit durchgeführt werden.
Prinzipielle Aussagen über das Verhalten eines DBMS können aber bereits durch Analyse des des Datenbanksystems zugrundeliegenden Prinzips vorgenommen werden.
Obwohl eine größere Zahl von Datenbankprinzipien existieren, so sollen im Folgenden nur die für technische Anwendungssysteme relevanten Prinzipien der Relationalen und der Objektorientierten Datenbanken untersucht werden.
Die Datenbankprinzipien der Hierarchischen Modells sowie des Netzwerkartigen Modells /194/ spielen in kommerziellen Anwendungssystemen eine signifikante Rolle, in technischen Problemstellungen ist ihr Einsatz selten.

4.1.1 Eigenschaften des Relationalen Modells und Relationaler DBMS

Das durch Codd /200/ eingeführte Relationale Modell hat seit seiner Einführung große Akzeptanz gefunden und verdrängt seither die bis dahin weitverbreiteten hierarchischen und Netzwerkmodelle. Die Grundlage des Relationalen Modells liegt in der Relationenalgebra begründet, die formal durch den Relationenkalkül beschrieben ist. Wesentliche Elemente der Relationenalgebra sind die Selektion (Auswahl von Tupeln

aus einer Relation), die Projektion (Auswahl von Spalten aus einer Tabelle), der natürliche Verbund (Verknüpfung zweier Relationen über allen gemeinsamen Attributen), die Mengenoperationen Vereinigung, Differenz und Durchschnitt sowie die Umbenennung.

Durch die Relationenalgebra bzw. den Relationenkalkül werden eine Reihe von Eigenschaften des Relationalen Modells vorgegeben. Diese Eigenschaften drücken sich einerseits durch strukturelle Eigenschaften der Anfragesprache oder durch Spezifika des DBMS selbst aus:

Eng verbunden mit der operationalen Eigenschaft des Relationalen Modells ist die Sprache SQL. SQL wurde zunächst als Anfragesprache (Data Manipulation Language, DML) konzipiert. Sie ist jedoch darüberhinaus auch als Datendefinitionssprache (Data Definition Language, DDL) erweitert worden.
SQL materialisiert die Eigenschaft einer **deskriptiven Anfragesprache**. Die Operanden von SQL sind Mengen von Tupeln. Die Behandlung dieser Mengen ist dabei ohne die Kontrollstrukturen konventioneller Programmiersprachen möglich. Im Gegensatz zu diesem deskriptiven (oder qualifizierenden) Verhalten von SQL stehen z.B. die navigierenden Anfragesprachen hierarchischer oder Netzwerk- oder objektorientierter Modelle /198/.

Das Prinzip der relationalen Operatoren ist die **Abgeschlossenheit**: Die Anwendung von Relationen Operatoren auf Relationen führt wieder zu Relationen. Die Optimierbarkeit von relationalen Anfragen basiert auf diesem Prinzip.

Die Forderung nach **Adäquatheit** besagt, daß die Prinzipien eines Modells durch die Anfragesprache ausgenutzt werden soll. SQL kann diese Adäquatheit nicht vollständig zugesprochen werden. Auf die Ergebnisse von SQL-Anfragen sind nicht alle Operationen (z.B. Selektionen auf Ergebnissen von Unionen) anwendbar.

Die Formulierung der Relationenalgebra ermöglicht die Anwendung der algebraischen Umformoperationen, die eine **Optimierung** eines Anfrageausdrucks erlauben. Das Ergebnis einer Anfrage darf dabei von der Optimierung nicht abhängig sein.

Strukturelle Ausprägung

Die Struktur des Relationalen Modells läßt sich verständlich in der sogenannten 3-Ebenen-Architektur nach ANSI/SPARC Abb. 22 darstellen. Die interne Ebene des Systems beschreibt dabei die implementierten Speicherstrukturen und Zugriffspfade. Die von systeminternen Spezifika abstrahierte konzeptuelle Ebene stellt für alle Anwender eine einheitliche Sicht auf den Inhalt der Datenbank dar. Die externe Ebene kann die für eine spezifische Anwendung relevante Sicht auf die definierten Daten darstellen.

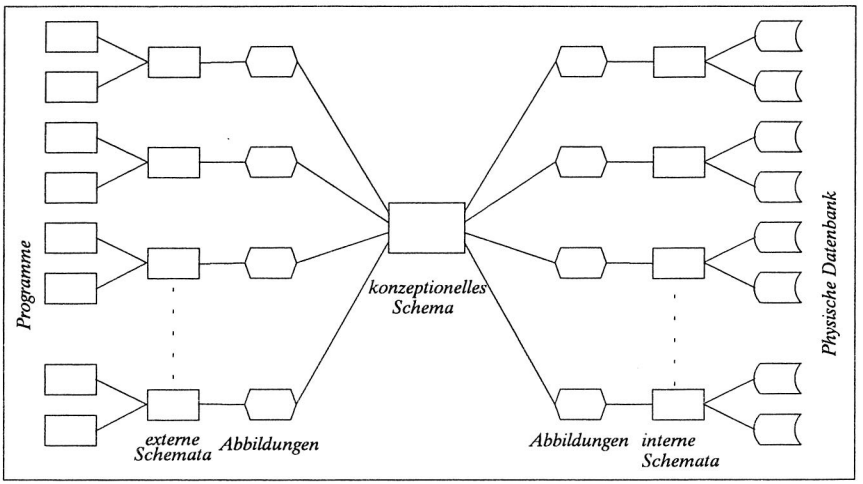


Abb. 22 3-Ebenen-Architektur von ANSI/SPARC nach /194/

Die Anlehnung des 3-Ebenen-Modells nach ANSI/SPARC mit der STEP-Entwicklungsmethodik (Kapitel 2.2) /83/ ist dabei äußerst vage. Die inhaltliche Verwandtschaft bezieht sich dabei hauptsächlich auf eine Dreiteilung der Entwurfsmethode allgemein.

Für den Entwurf von Datenbanken, die in einem DBMS relationaler Prägung verwaltet werden sollen, haben sich eine Reihe von Entwurfsmethoden etabliert, die für einen realen Sachverhalt ein Modell erstellen sollen, welches in einer Datenbank implementiert werden kann. Zu unterscheiden sind hierbei Dekompositions- (z.B. 3NF-Dekompositionsverfahren von Codd /200/, 4Fn-Dekompositionsverfahren von Lien) und Synthesealgorithmen (Synthesealgorithmus von Bernstein oder Meier). Wesentliche Einfluß auf die Definition eines Datenbankschemas hat die Normalformenlehre genommen, die in beiden Verfahren zu einer Bewertung der Entwurfsqualität herangezogen werden kann.

Defizite des Relationalen Modells

Die Defizite des Relationalen Modells lassen sich nach /198/ auf die Teilbereiche

- Datenmodellierung
- Datenbankentwurf
- Anfragesprachen
- Update-Operationen
- Zugriffspfade und
- Optimierung

konkretisieren. Im folgenden werden kurz die wesentlichen Probleme des relationalen Modells erläutert. Für die im weiteren wichtigen Problembereiche der Abbildbarkeit von EXPRESS-Konstrukten auf Relationale Modells werden in Kapitel 4.1.3 spezifische Defizite angesprochen.

Der Begriff der *Datenmodellierung* umfaßt in dem hier benutzten Rahmen die Möglichkeiten der Modellerstellung mit Hilfe der vom Modellierungsparadigma bereitgestellten Prinzipien bzw. Konstrukte. Die Verfügbarkeit ausdrucksstarker Modellierungskonstrukte und deren Unterstützung in Anfrage- oder Update-Operationen spielt hierbei eine wesentliche Rolle.

Die *Normalformenlehre* des Entwurfs Relationaler Datenbanken führt zu einer Verteilung von Attributen auf Relationen, die unter dem Einfluß formaler Kriterien erfolgt. Die Semantik des darzustellenden Modells geht somit in einer Anzahl einzelner Relationen auf. Unterschiedliche Beziehungen wie "B ist ein A", "B ist eine Komponente von A" oder die Darstellung mengenwertiger Attribute werden im Relationenmodell durch Fremdschlüssel-Beziehungen in den Relationen dargestellt.

Die Begrenztheit der Leistungsfähigkeit der *Anfragesprachen* relationaler Datenbanken läßt sich durch die Einbettung in eine höhere Programmiersprache lösen. Nach Codd (Co82) soll die Anfragesprache eines relationalen Datenbanksystems *relational vollständig* sein, also zumindest das leisten, was die Relationenalgebra bzw. das Relationenkalkül vermag, und dies mit *rein deskriptiven Mitteln*, also ohne algorithmische Sprachkonstrukte. Die Bestimmung von Größen, die aus Daten der Datenbank auf komplexere Art berechnet werden müssen, erfordert daher die Einbindung einer Datenbankanfrage in eine Programmiersprache. Die Problematik der Einbettung einer Anfragesprache einer relationalen Datenbank in eine Programmiersprache ruft das Problem des *impedance mismatch* hervor. Dies ist das Nichtzusammenpassen der unterschiedlichen Paradigmen der kalkülartigen Anfragesprache mit der imperativen Programmiersprache.

Damit zusammenhängend ist das Problem der schwachen *Typisierung* des relationalen Modells zu nennen. Klassische relationale Systeme unterstützen nur einfache Datentypen wie INTEGER, STRING, DATUM und MONEY. Dies ist durch die Forderung nach Unterstützung nur *generischer Operationen* auf den Datenbankbestand begründet. Da die Werte von Tupelattributen nur als abstrakte Symbole gespeichert werden, sind damit nur einfache Selektionsbedingungen wie Identität oder bei Definition einer Ordnung von Werten auch eine Ordnungsbedingung formulierbar. Die anwendungsorientierte Nutzbarkeit von Daten ist jedoch auf die operationale Verwertbarkeit der Daten gebunden.

Die bisher angesprochenen Probleme liegen im Bereich der strukturellen Ausprägung einer Datenbank. Die Problematik von Updateoperationen auf den Datenbestand einer Datenbank sind durch die Eigenschaft der Tupelorientiertheit, der Angewiesenheit auf

explizite Integritätsbedingungen, der Identifikation von Objekten über sichtbare Schlüssel und des Fehlens objektspezifischer Updates begründet.

Da komplexe Objekte einer Anwendung in einer relationalen Datenbank über mehrere Relationen verteilt werden, eine Änderungsoperation in einer relationalen Datenbank aber meist nur die Daten in einer Relation verändern kann ergeben sich folgende zusätzlichen Probleme: Die Identifikation des zu ändernden Datensatzes kann durch direkte Bezeichnung des Datensatzes erreicht werden, sondern nur durch eine deskriptive Beschreibung, in SQL etwa durch eine "SELECT ... WHERE .." Anweisung. Damit ist der Zugriff auf die gesamten Datensatz eines komplexen Objektes nicht strukturell möglich. Werden durch eine Selektionsbedingung eine Menge von Tupeln erreicht, so gelten die Änderungsanweisungen homogen für alle selektierten Tupel. Die selektive Änderung eines Tupels eines komplexen Objektes ist dann nicht möglich.

Während das Relationale Modell ein einheitliches Verständnis über Relationale DBMS geschaffen hat und damit wesentliche Eigenschaften von Datenbanksystemen auf dieses Modell zurückzuführen sind, liegt die große Schwäche des relationalen Modells u.a. in der Begrenztheit der Ausdrucksmöglichkeiten des relationalen Prinzips.
Das im folgenden vorgestellte Prinzip objektorientierter DBMS kann zumindest dieses Defizit relationaler Modelle lösen.

4.1.2 Eigenschaften Objektorientierter DBMS

Während das relationale Modell und die darauf aufbauenden Systeme relationaler Datenbanken eine Akzeptanz in vielen Anwendungsbereichen erfahren haben, besteht eine einheitliche Vorstellung über ein objektorientiertes Datenbankmodell noch nicht. Die Verbreitung von objektorientierten Datenbanksystemen wird offensichtlich auch hierdurch gebremst.
Bedingt durch die weitverbreiteten Anwendung objektorientierter Programmiersprachen und der Anforderung, die in den objektorientiert realisierten Anwendungssystemen vorhandenen Daten persistent verwalten zu können, entstanden Objektorientierte Datenbanken. Die im Vergleich zu relationalen oder anderen Datenbankprinzipien relativ junge Geschichte objektorientierter Datenbanken hat ein einheitliches Verständnis dieses Datenbankprinzips noch nicht durchzusetzen vermocht. In /199/ wird versucht, ein einheitliches Verständnis für das objektorientierte Datenbankmodell zu schaffen. Die Autoren unterscheiden Kriterien, die für ein objektorientiertes Datenbankmodell obligatorisch und solchen, die optional sind. Darüberhinaus stellen sie implementierungsabhängige Entwurfsentscheidungen objektorientierter Datenbanksysteme zusammen:

Zu den fünf notwendigen Eigenschaften objektorientierte Datenbanksysteme gehören:

- *Komplexe Objekte:* Das Datenbanksystem muß neben einfachen (atomaren) Typen wie Integer, Real, String usw. die Darstellung komplexer Objekte erlauben. Diese

komplexen Objekte müssen über Konstruktoren zu komplexen Objekten wie Tupeln, Mengen (Set), Listen (List) oder Felder (Array) zusammengesetzt werden können.

- *Objektidentität:* Ein objektorientiertes Datenbanksystem muß die Identität eines Objektes, oder besser eines Wertes dieses Objektes sicherstellen. Zwei Objekte können gleich sein, falls sie den selben Wert besitzen oder identisch sein, falls es sich um das selbe Objekt handelt.
- *Kapselung:* Das Prinzip der Kapselung verlangt die Trennung von Spezifikation und Implementierung eines Objektes.
- *Typen und Klassen:* Nach Ansicht der Autoren sind Typen Hilfsmittel zur Benennung von Dingen, die ein Programmierer manipuliert. Sie dienen zur Überprüfungen der Korrektheit seines Programms zur Kompilierzeit. Im Gegensatz dazu werden Objekte hier als ein dynamischer Typisierungsmechanismus verstanden, der die Typisierung zur Laufzeit bewirkt. Sie dienen der Unterstützung der Manipulation von Objekten.
- *Klassen- oder Typhierarchie:* Das Prinzip der Klassen- oder Typhierarchie erlaubt die Sammlung von Objekten in generalisierenden bzw. spezialisierenden Trägern. Damit wird die Vererbung von Eigenschaften entlang der Hierarchie möglich.
- *Redefinition, Überladen und spätes Binden:* Spezialisierten Objekten werden andere (speziellere) Methoden zugeordnet (overriding), die jedoch unter einer einheitlichen Bezeichnung verfügbar sind (overloading). Dieses Prinzip wird als Polymorphismus bezeichnet.
- *Berechnungsvollständigkeit*: Die Datenmanipulationssprache (DML) der Datenbank muß berechnungsvollständig sein, was bedeutet, daß sich in ihr jede berechenbare Funktion ausdrücken lassen muß. Die DML der relationalen Welt SQL ist nicht berechnungsvollständig.
- *Erweiterbarkeit:* Die Datenbank besitzt vordefinierte Typen. Neben diesen Typen muß die Möglichkeit für den Anwendungsprogrammierer bestehen, eigene Typen zu definieren, die sich für ihn von den vordefinierten nicht unterscheiden.

Die vorgenannten Eigenschaften an Datenbanksysteme entstammen den Prinzipien objektorientierter Programmiersprachen. Die folgenden Anforderungen sind typische Eigenschaften herkömmlicher, insbesondere relationaler Datenbanken.

- *Persistenz:* Persistenz ist die Fähigkeit, Daten eines Anwendungsprogramms über die Dauer der Ausführungszeit eines Programms hinweg zur Verfügung zu haben. Die Persistenz OODBS soll orthogonal zu den Klassen und Typen sowie für den Programmierer transparent sein.
- *Sekundärspeicherverwaltung:* Die Fähigkeit große Datenmengen halten und verwalten zu können wird durch Mechanismen wie Indexverwaltung, Datenpufferung, Zugriffspfadverwaltung und Anfrageoptimierung erreicht.

- *Mehrbenutzerbetrieb:* Hierbei soll ein OODBMS die Fähigkeiten zur Verfügung stellen, wie sie von den verfügbaren (relationalen) Systemen geboten werden. Hierunter fallen z.B. Atomarität von Operationen und Serialisierbarkeit.
- *Recoveryfähigkeit:* Ein OODBMS soll analog den Fähigkeiten verfügbarer Datenbanken ein Recovery nach Soft- und Hardwarefehlern ermöglichen.
- *Ad-Hoc-Anfragesprachen:* Neben der eigentlichen Fähigkeit, einen Datenzugriff über eine textuelle Anfrage zu ermöglichen, genügt den Autoren hier auch die Möglichkeit, den Datenbestand über graphische Editoren (Browser) zu durchsuchen.

Neben diesen notwendigen Eigenschaften zählen die Autoren auch fünf optionale Eigenschaften wie Mehrfachvererbung, Typüberprüfung, Unterstützung verteilter Systeme und lange Transaktionen.
Die ersten acht Eigenschaften werden oftmals von den objektorientierten Konzepten der der Datenbank zugrundeliegenden Programmiersprachen der objektorientierten Datenbanken bereitgestellt und damit von vielen Systemen ausreichend unterstützt. Die Anforderungen, die den konventionellen Datenbanken entnommen wurden, sind eher den funktional bereitgestellten Fähigkeiten relationaler Systeme nachempfunden. Hier besitzen OODBMS noch teilweise den konventionellen, relationalen Systemen unterlegene Fähigkeiten.

Aufgrund des identifizierten Defizits bei der Standardisierung objektorientierter Datenbanken wurde von den Anbietern objektorientierte Datenbanken eine ad-hoc Arbeitsgruppe zur Spezifikation eines Standardisierungsvorschlages gebildet (ODMG Object Database Management Group). Der Standardisierungsvorschlag dieser Arbeitsgruppe umfaßt ein Objektmodell, eine Objektdefinitionssprache ODL (Object Definition Language), eine Anfragesprache OQL (Object Query Language) sowie Sprachanbindungen für C++ und Smalltalk.
Ob die Definition der ODMG einen ähnlichen Durchbruch für die Anwendung objektorientierter Datenbanken wie SQL und das relationale Modell für Relationale Datenbanksysteme bewirkt hat bleibt fraglich, jedoch wird hiermit zunächst das Verständnis für Objektorientierte Datenbanken vereinheitlicht.

4.1.3 Abbildbarkeit EXPRESS-definierter Schemata auf Datenbankmodelle

Die Realisierbarkeit von STEP-Datenbanken ist im wesentlichen gekennzeichnet durch die Fähigkeit eines Datenbankmodells, Konstrukte der Sprache EXPRESS günstig auf Konzepte der Datenbank abbilden zu können. Da die Zahl von STEP-Anwendungsprotokollen nicht begrenzt ist und damit die Menge der benutzten EXPRESS-Konstrukte prinzipiell nicht beschränkt ist, muß die Realisierbarkeit einer STEP-Datenbank mit der Realisierung beliebiger Datenbankschemata nach vorgegebener EXPRESS Schemabeschreibung verallgemeinert werden. Die Realisierung einer STEP-Datenbank läßt sich also auf die Abbildbarkeit von EXPRESS-Sprachkonstrukten konkretisieren.

Das in den STEP-Anwendungsprotokollen entwickelte Datenmodell stellt das Ergebnis eines konzeptionellen Datenbankentwurfes dar. Offensichtlich ist dieses Datenmodell nicht normalisiert /194/, was für die Überführung in ein relationales Modell gefordert wird. Die Rückführung eines STEP-Datenmodells wird in /124/ angestrebt, von anderer Seite jedoch aus Gründen der einfacheren, direkten Übertragbarkeit unterlassen.

Neben einigen ad-hoc Realisierungen von STEP/EXPRESS Datenbanken /187//188/, wurde in /186/ die Realisierbarkeit solcher Datenbanken systematisch untersucht. Dabei wurde primär auf die oben erwähnte Abbildbarkeit von EXPRESS-Sprachkonstrukten auf Datenbankprinzipien eingegangen. Weitere Kriterien wie Performanz oder Entwicklungs- und Anwenderunterstützung wurden dabei vernachlässigt.

Die Ergebnisse der theoretischen Untersuchungen sollen im folgenden kurz zusammengefaßt werden. In der Übersicht Abb. 23 wird qualitativ ersichtlich, welche Konzepte bzw. welche Prinzipien mit welchem Datenbanksystem effizient zu realisieren sind.

Konzept \ *DBMS*	*Sybase*	*Ingres+*	*Postgres*	*ObjectStore*	*Ontos*	*Gemstone*
Entities						
Vererbung						
Objektidentität						
Typen						
Algorithmen						
Integritätsbedingungen						

Abb. 23 Unterstützung der EXPRESS-Konzepte auf verschiedenen Datenbanksystemen

Die relationalen Datenbanksysteme Sybase und Ingres haben vor allem mit Defiziten des relationalen Modells zu kämpfen, wie sie in Kapitel 4.1.1 skizziert wurden. Für Gemstone, einem auf dem Sprachmodell von Smalltalk basierenden Datenbanksystem, ist vor allem die aus Smalltalk bekannte Einschränkung auf einfache Vererbung sowie dem nachteiligen Antwortzeitverhalten charakteristisch. Die wesentlichen Schematransformationsmechanismen und deren Einschränkungen sind aufgrund veröffentlichter Implementierungsversuche in Kapitel 4.2 dargestellt.

Die objektorientierten Datenbanksysteme ObjectStore und Ontos mit den Implementierungssprachen C++ und Gemstone mit seiner Implementierungssprache Smalltalk haben aufgrund der Ähnlichkeit ihrer Modellparadigmen mit der von EXPRESS deutliche Vorteile in der Transformationsproblematik von EXPRESS-Modellen in das Daten-

banksystem. Die verbleibenden Probleme liegen zumeist im unterschiedlichen Verständnis der Entity bzw. Objektbegriffes bzw. der in EXPRESS äußerst differenziert formulierbaren Vererbungsbeziehungen und den sich hieraus ergebenden Unterschiede. Hierzu wird in Kapitel 4.3 näher eingegangen.

4.2 Realisierungen von STEP-Datenbanken auf Relationalen DBMS

Die für Relationale DBMS fundamentale Schnittstelle wird durch die normierte Anfragesprache SQL gebildet /206/. Bezeichnend für die Abbildung von EXPRESS-Schemata auf Relationale DBMS ist neben des Funktionsumfanges des DBMS als solches, der Umfang der von dem spezifischen System unterstützten SQL-Dialekte/207//9/.
Die Realisierung einer STEP-Datenbank mit einem RDBMS als Basissystem wurde erstmals in /187/ erwähnt. Hierauf aufbauend wurde in /188/ ein Schemaübersetzer entwikkelt, der SQL89 Kommandos für das RDBMS ORACLE erzeugt. In /124/ wird eine Abbildung von EXPRESS auf den Sprachumfang von SQL2 diskutiert.
In /186/ werden Realisierungsalternativen auf INFORMIX und INGRES gegenübergestellt und mit den bekannten Alternativen diskutiert.

Der Abbildungsmechanismus eines vorliegenden Schemas auf ein Relationales DBMS gehorcht in diesen Fällen meist ähnlichen Gesetzen, die im Folgenden kurz dargestellt werden: Entities werden als Tabelle abgebildet. Die in einem Entity definierten EXPRESS-Attribute werden auf Attribute (Felder) der Tabelle abgebildet. Instanzen der Entitäten lassen dann als Tupel in die Relation eintragen. Die Problematik der (Instanz-) Objektidentität kann in reinen SQL-Datenbanken nicht gelöst werden, da eine Tupelidentität von diesem Datenbankprinzip prinzipiell nicht unterstützt wird. In /124/ wird die datenbankspezifischen Erweiterung einer Zeilenidentifikation (*rowid*) und in /188/ ein selbstdefiniertes Surrogat aus dem Entitynamen und einer systemweit eindeutigen Zahl als Tupelidentifikator benutzt.

Die Problematik der Bezeichner für Entities bzw. Tabellen und Attribute bzw. Felder stellt ein weiteres Problem bei der Abbildung in relationale Datenbanken dar: EXPRESS erlaubt beliebig lange Zeichenketten für alle Bezeichner, während Datenbanksysteme diese auf Längen von beispielsweise 18 (INFORMIX) oder 32 (INGRES) Zeichen beschränken. Der prinzipielle Lösungsweg hierfür ist die Abbildung langer Namen auf Kurznamen, wobei aber zum Zugriff auf die Anwendungsdaten eine Tabelle dieser Transformation selbst in der Datenbank abzuspeichern ist /188/.

Das EXPRESS-Typsystem ermöglicht eine ausführliche Typisierung, die in Relationalen Datenbanksystemen in dieser Reichhaltigkeit nicht unterstützt wird. Für die einfachen EXPRESS-Datentypen INTEGER, REAL und NUMBER stehen zumeist akzeptable eigene Datentypen wie integer, float und decimal(n) in SQL2 gegenüber. Die Einschränkung der Wertbegrenzung dieser Datentypen gegenüber den unbeschränkten EX-

PRESS-Datentypen wird durch die Anbindung eines Anwendungssystems über eine Sprachanbindung eines SDAI relativiert.

Der EXPRESS Typ STRING erlaubt die Darstellung beliebig langer Zeichenketten und stellt damit ein weiteres Problematik der Abbildung dar. Die in SQL2 verfügbaren Typen char(n) und varchar(n) stellen Zeichenketten fester bzw. variabler Länge zur Verfügung. Die Problematik liegt jedoch in einer begrenzten Maximalgröße eines Stringwertes (z.B. SQL2 32kB) bzw. eines Tupels (z.B. INGRES 2kB). Moderne Datenbankerweiterungen bieten einen Datentyp Text an, der als sogenanntes BLOB (Binary Large Object) außerhalb der Datenbank realisiert wird. Für Ausprägungen dieses Datentyps können aber oftmals nicht alle Datenbankoperationen angewandt werden.

EXPRESS-Konstrukt	*SQL – Konstrukt*
INTEGER	*integer*
REAL	*float*
NUMBER	*decimal*
STRING	*char/varchar*
BINARY	*binary*
LOGICAL/BOOLEAN	*integer*
ARRAY/BAG/LIST/SET	*Relationen für Typ und Elemente*
ENTITY	*Relation*
TYPE	*Basistyp der Typdefinition*
ENUMERATION	*Eine Relation für alle Aufzählypen oder eine Relation für jeden Aufzähltyp*
SELECT	*Relation*
DERIVE	*Trigger und Datenbankprozeduren (SQL-Erweiterungen)*
RULE/WHERE	*check, Datenbanprozeduren (SQL-Erweiterungen)*
UNIQUE	*unique*
Supertyp/Subtyp-Hierarchie	*keine direkte Abbildung: Verbindung von Relationen über Identifikator oder Vererbungsbeziehung über eigene Relationen Integration der vererbten Attribute in Supertyp- oder Subtyprelation*

Tab. 4 Abbildungsmechanismen von EXPRESS-Konstrukten in SQL-basierende Relationale DBMS

Wertausprägungen der Datentypen BOOLEAN und LOGICAL, die von SQL2 nicht unterstützt werden lassen sich einfach als Festkommazahlen abbilden /188/. Die Abbildung von Aufzähltypen läßt sich über eine eigene Relation für jeden Aufzähltyp /187/ oder eine Relation für alle Aufzähltypen /187/ realisieren. Die Abbildung der EXPRESS Auswahltypen (SELECT) wird in der Literatur ähnlich behandelt. Jedoch erschwert die Tatsache, daß nicht nur Entities Elemente des Auswahltyps sein können die Abbildung. In /186/ wird eine erweiterte Relation vorgeschlagen, die für jeden möglichen Basisty-

pen der Auswahlliste ein Feld bereithält, und in einem weiteren Feld die Ausprägung des aktuellen Wertes beschreibt.

Aggregierte EXPRESS Datentypen erzwingen ebenfalls die Auslagerung auf zusätzliche Relationen. Die Tabelle enthält dann zumindest ein Feld, das vom Basistyp des Aggregates definiert wird. Die Werte der Komponenten der Aggregate bilden dann die Tupel dieser Tabelle. Damit lassen sich ebenfalls geschachtelte Aggregate bilden. Die für die unterschiedlichen Aggregattypen notwendigen Zusatzinformationen müssen den jeweiligen Basistabellen zugeordnet werden /186/.

Die Berechnung abgeleiteter Attribute in einer Relationalen Datenbank kann über SQL2 noch nicht formuliert werden. Spezielle Erweiterungen von Datenbanksystemen erlauben über Trigger die Initiierung von Berechnungen, die über in SQL zu definierende Ausdrücke bestimmt werden.

Die durch den Supertyp/Subtyp-Graphen gebildete Vererbungshierarchie der EXPRESS Entitäten kann auf unterschiedliche Methoden auf Relationen abgebildet werden:
/187/ verwendet den Ansatz, sowohl Supertypen als auch Subtypen in jeweils eigene Relationen abzubilden. Diese werden über einen Identifikator verbunden. Die Supertyp-Relation erhält einen Diskriminator, um zwischen den Ausprägungen unterschiedlicher Subtypen unterscheiden zu können.
Eine Abwandlung dieses Prinzips modelliert die Vererbungsbeziehung außerhalb der Typrelationen in eigenen Relationen.
/188/ fügt die Attribute der Supertypen den Subtypen hinzu und bildet nur die Subtypen als eigene Relationen ab. Für die Supertypen werden darüberhinaus Views erzeugt. Weiterhin lassen sich alle Subtypen innerhalb des Supertyps einordnen. Ein große Anzahl leerer Felder spricht jedoch gegen dieses Prinzip.

Die in EXPRESS Schemata definierbaren Integritätsbedingungen wie lokale und globale Regeln sowie UNIQUE-Klauseln lassen sich mit Relationalen Datenbanken teilweise einfach lösen: Im Fall der UNIQUE-Klausel, die direkt als SQL UNIQUE abgebildet werden kann (hingegen lassen sich EXPRESS UNIQUE Klauseln auf abgeleitete oder aggregierte Werte in SQL nicht formulieren), bereitet die Abbildung der in EXPRESS beliebig komplex definierbaren Regeln größere Schwierigkeiten: Einerseits kann durch die SQL CHECK eine Wertebereichsüberprüfung realisiert werden, mit Datenbanktriggern und Datenbankprozeduren können andererseits auch komplexere Berechnungen durchgeführt werden.

4.3 Realisierungen von STEP-Datenbanken auf OODBMS

Die Realisierung von STEP-Datenbanken auf objektorientierten Datenbanken ist historisch gesehen wesentlich jünger als die ersten Implementierungen auf Relationalen Da-

tenbanksystemen (siehe Kapitel 4.2). Trotz der jungen Geschichte von OODBMS wurden einige Implementierungsversuche in diesem Bereich bekannt. Das Interesse an Implementierungen von STEP-Datenbanken auf OODBMS ist offensichtlich durch die stärkere konzeptionelle Ähnlichkeit des objektorientierten Prinzips mit den Eigenschaften von EXPRESS zurückzuführen. Implementierer erhoffen sich hiervon einfachere Abbildungsmechanismen zwischen Konzepten von EXPRESS und dem zugrundeliegenden Datenbanksystem.

Die Implementierung von STEP-Datenbanken auf objektorientierten Datenbanksystem ist im Gegensatz zu Implementierungen auf Relationalen Datenbanken wesentlich stärker durch das der Datenbank zugrundeliegenden Sprachmodell determiniert. In OODBMS, die die Persistenz von Variablen (Instanzen) von C++ Anwendungen unterstützen (z.B. ObjectStore, Ontos) kann die Abbildung von EXPRESS-Modellen nahezu mit der Abbildung in die Basissprache des Datenbanksystems (also z.B. C++) gleichgesetzt werden.
Die Realisierung von STEP-Datenbanken auf objektorientierten Datenbanken ist damit aber wesentlich stärker an die Anbindung einer bestimmten SDAI-Realisierung gebunden, als dies für Implementierungen auf Relationalen Datenbanken gilt. Der Vorteil der objektorientierten Datenbanken, ein äquivalentes Sprachmodell wie ein Anwendungssystem auf der STEP-Datenbank zu besitzen darf nicht durch ein Modellmismatch zwischen Datenbankimplementierung, SDAI-Implementierung und Anwendung konterkariert werden. Die Realisierung einer Abbildung von EXPRESS-Modellen auf Objektorientierte Datenbanksysteme sollte sich daher eng an die Spezifikation der C++-Sprachanbindung des SDAI halten. Abbildungen von EXPRESS-Modellen auf Objektorientierten Datenbanken, die über eine SDAI-C-Sprachanbindung verfügen sollen, würden die Fähigkeiten des objektorientierten Prinzips weitestgehend verschenken. Solche Realisierungen sind in der Literatur nicht bekannt.

Auf die Abbildung eines EXPRESS-Modells soll hier nicht weiter eingegangen werden. Die Mechanismen hierfür werden in Kapitel 4.5.1 weiter ausgeführt.

In /186/ werden die Abbildungsmechanismen für EXPRESS auf das OODBMS ObjectStore vorgestellt. In diesem Zusammenhang wird der Unterschied zu der ebenfalls gezeigten Abbildung auf Relationale Datenbanksysteme, insbesondere SQL-basierte Datenbanken hervorgehoben. Der Abbildungsmechanismus orientiert sich dabei an der C++-Sprachanbindung des SDAI /45/.
In /103/ wird für eine frühere SDAI-Spezifikation die automatisierte Übersetzung von EXPRESS-Modellen in Datenbankschemata für ObjectStore beschrieben. Dabei wird die Eigenschaft von ObjectStore benutzt, C++-Anwendungsklassen durch Einbindung in die Datenbank Persistenz zu verleihen. Die Übersetzung benutzt dabei die C++-Klassengenerierung des NIST-Toolkits /201/.

Mit laufender Konkretisierung der SDAI-Spezifikation haben auch kommerzielle Anbieter bereits erste Implementierungen von STEP-Datenbanken auf objektorientierten Datenbanksystemen realisiert. Die dabei erzielten Ergebnisse stehen in direkter Abhän-

gigkeit mit dem Konkretisierungsgrad des Normdokumentes zur C++-Sprachanpassung. Mit Standardisierung des Dokumentes werden sicherlich weitere Anbieter folgen.

4.4 Implementierung einer STEP-Datenbank auf dem Erweitert Relationalen Datenbankmanagementsystem Postgres

Wie in Kapitel 4.1.3 kurz angedeutet, hat sich das DBMS Postgres mit seinem Erweitert Relationalen Datenbankmodell als prinzipiell mächtig genug erwiesen, um eine effiziente STEP/EXPRESS Datenbank zu unterstützen. Für die Auswahl einer Implementierung einer STEP/EXPRESS-Datenbank spielt jedoch nicht nur die Fähigkeit der effizienten Schematransformation eine Rolle. Ein weiteres Kriterium das zur Auswahl von Postgres als das Basissystem führte, war die Tatsache, daß Postgres eine auf mehreren Ebenen mögliche Erweiterbarkeit besitzt, sowie die Verfügbarkeit des Systems in Quellcode. Die Verfügbarkeit des Quellcodes ermöglicht Fehlerbereinigungen aus eigener Kraft.

Im folgenden soll zunächst das Modell von Postgres erläutert werden. dabei sollen besonders die Konzepte im Vordergrund stehen, die die Implementierbarkeit einer STEP/EXPRESS Datenbank beeinflussen.
Im weiteren wird explizit die Wahl des Transformationsverfahrens von EXPRESS-Konstrukten auf Postgres-Konzepte erläutert. Dabei wird vor allem die Abbildung der modellstatischen Konstrukte von EXPRESS eingegangen. Die Grenzen der Abbildung lassen sich dabei darlegen.

Für die Implementierung einer STEP-Datenbank gemäß einem STEP Anwendungsprotokoll müßten nach Festlegung des Abbildungsschemas alle innerhalb des EXPRESS Schemas definierten Konstrukte in das Datenbankschema überführt werden. Eine manuelle Implementierung der EXPRESS-Schemakonstrukte in die Datenbank würde einen hohen Aufwand bedeuten. Aus Gründen der Definierbarkeit formaler Transformationsregeln bietet sich jedoch eine automatische Wandlung des EXPRESS-Schemas in ein Datenbankschema an. In Kapitel 4.4.4 wird auf eine derartige automatisierte Wandlung eingegangen.

4.4.1 Erweiterungen des Postgres-Datenmodells zum Relationalen Modell

Das Datenmodell von Postgres basiert auf Erweiterungen des relationalen Modells wie es von Codd eingeführt wurde, und unterstützt somit prinzipiell alle Konzepte, die auch von anderen relationalen DBMS unterstützt werden. Darüberhinaus besitzt Postgres Mechanismen, die eine semantische Modellbildung erlauben.

Die strukturellen Eigenschaften von Postgres, die sich von der relationaler Systeme unterscheiden, sind das Typsystem, die Objektidentität, Vererbung und Integritätsbedingungen.

Als Typ eines Relationenattributs lassen sich in Postgres eine Vielzahl von "internen" Typen wie diverse Festkommazahlen, Gleitkommazahlen, Strings unterschiedlicher, auch variabler Länge aber auch komplexe Typen wie das zweidimensionale Rechteck (box), das zweidimensionale Polygon (polygon) usw. In Postgres werden Typkonstruktoren über beliebigdimensionale Arrays von beliebigen Basistypen unterstützt. Mit Hilfe eindeutiger Objektidentifikator (oid) können Tupel als Attribute referenziert werden.

Die Unterstützung benutzerdefinierter Typen muß über die Bereitstellung von Konvertierungsfunktionen zur Umwandlungen von Werten des Typs von der internen Repräsentation zu einer Stringdarstellung und umgekehrt erfolgen. Diese Funktionen müssen in C implementiert werden und werden nach der Deklaration des Typs dynamisch in die Datenbank geladen. Danach ist der Typ in der Datenbank verfügbar und unterscheidet sich in keiner Weise von den systemeigenen Typen. Tatsächlich sind die Postgres inhärenten Typen durch ein analoges Vorgehen beim Installieren der Datenbank eingebracht worden.

Postgres unterstützt das Prinzip der Objektidentität durch einen vom System vergebenes, datenbankweit eindeutigen Objektidentifikator (oid). Dieser wird durch ein zusätzliches, nicht veränderbares Attribut von Typ einer Festkommazahl ermöglicht.

Mit Hilfe des Vererbungsprinzipes ermöglicht Postgres die Modellierung einer Typhierarchie. Postgres erlaubt mehrfache und tiefe Vererbung. Es existiert jedoch kein Mechanismus zum Auflösen von Vererbungskonflikten bei mehrfacher Vererbung eines Attributs mit identischem Namen.

Zur Einhaltung von Integritätsbedingungen können in Postgres Regeln definiert werden. Postgres Regeln halten sich an die klassische ECA-Semantik (Event-Condition-Action). Im Bedingungsteil der Regel kann auf die Postgres Aktionen Zugriff (retrieve), Update (replace), Löschen (delete) und Erzeugen (append), die auf ein Postgres Tupel wirkt, angewandt werden. Der Aktionsteil gibt die Möglichkeit, in Postquel formulierte Anweisungen zu definieren, die entweder zusätzlich zu der im Bedingungsteil angegebenen Aktion oder anstatt dieser (*instead*) ausgeführt wird.
Mit Hilfe der Postgres Regeln läßt sich die referentielle Integrität eines Anwendungsmodells sicherstellen. Darüberhinaus sind in Postgres Sichten (Views) über ein "Query-Rewrite System" unter Benutzung der Postgres Regeln implementiert.

Die Erweiterung der Postgres Sprache (DDL und DML) Postquel durch Operationen, die den indizierten Zugriff auf Arrays, auf Komponenten eines Objektes sowie den Aufruf von Funktionen erlaubt stellt eine kompatible Obermenge von QUEL dar. Postgres erlaubt die Formulierung von unären und binären Operatoren, wobei erstere als linksunäre oder rechtsunäre Operatoren definiert werden können. Die Implementierung eines

benutzerdefinierten Operators muß in einer Postgres Programmiersprache erfolgen. Das Ergebnis einer Anfrage ist in Postgres als View zu verstehen, auf das im Gegensatz zu relationalen Systemen sogar Updates wirken können.

Postgres unterstützt zwei verschiedene Arten von Funktionen. Einerseits sind dies Funktionen, die in der Postgres Anfragesprache Postquel definiert sind, andererseits lassen sich Funktionen auch in konventionellen Programmiersprachen realisieren, solange sie sich an Bindungskonventionen der Implementierungssprache von Postgres C halten. Beide Arten von Funktionen können als Parameter systemdefinierte Basistypen oder benutzerdefinierte Typen als Parameter aufnehmen. Darüberhinaus können sie Werte eines beliebig definierten Typs zurückliefern.
Postquel Funktionen können im Funktionsrumpf alle beliebigen Postquel Kommandos aufnehmen, sind aber dabei auf Postgres-Funktionalität beschränkt.
Die interessante Alternative für Funktionsimplementierung in Postgres sind Funktionen, die in Programmiersprachen realisiert werden, die die Integration in das Datenbanksystem selbst erlauben. Diese Funktionen werden dann zur Laufzeit in das DBMS geladen und ausgeführt. Das Laden der Funktion erfolgt über dynamisches Binden von "shared Libraries", wozu die Funktionalität des dynamischen Bindens des Betriebssystems benutzt wird. Zu Testzwecken bei der Implementierung benutzerdefinierter Funktionen können diese Funktionen auch in einem getrennten Adreßraum ausgeführt werden, um bei einem induzierten Programmfehler und einem Absturz des DBMS den Datenbankinhalt nicht zu zerstören.

Das Prinzip der Erweiterbarkeit von Postgres besteht aus der Offenheit des datenbankeigenen Systemkatalogs. Der Unterschied zu konventionellen relationalen Datenbankmanagementsystemen besteht darin, daß Postgres weitaus mehr Information über die Datenbank und das DBMS speichert. Relationale System speichern prinzipiell nur Informationen über Relationen und Tupel im Systemkatalog. Postgres hingegen speichert darüberhinaus auch Typinformationen, Informationen über Funktionen, Zugriffsmethoden usw. Das DBMS Postgres orientiert sich im Betrieb wesentlich stärker an den Systemkatalog als vergleichbare relationale Systeme. Durch die Veränderbarkeit des Systemkatalogs auch von Benutzerseite ist die Erweiterbarkeit von Postgres zu erklären. Durch Integration von benutzerdefinierten Funktionen ist darüberhinaus eine weitgehende Erweiterbarkeit der Datenbankfunktionalität möglich.

Die hier kurz beschriebenen Möglichkeiten von Postgres geben den Ausschlag, dieses DBMS als Basis einer Produktdatenbank zu nutzen, deren Datenmodell einem in der ISO 10303 festgelegten Produktmodell festgeschrieben ist. In Kapitel 4.4.3 werden die speziellen Abbildungsmechanismen von EXPRESS Datenmodellen nach einem Postgres Datenbankschema dargelegt.

4.4.2 Postgres als STEP-Produktdatenbank

Für die Nutzung eines Datenbestandes mit Hilfe einer Datenbank muß dieser Datenbestand in einer festgelegten Form in die Datenbank eingebracht und über eine festge-

legte Schnittstelle den Anwendungssystemen angeboten werden. Die Vorgehensweise des Datenbankentwurfes zur Erstellung eines Datenmodelles ist nach ANSI/SPARC /209/ in einem Referenzmodell festgelegt und beschreibt darin die prinzipielle Vorgehensweise.

Eine STEP-Produktdatenbank muß Produktdaten in einem Datenmodell ablegen und Anwendungssystemen zur Verfügung stellen, das in einem (oder mehreren) STEP-Anwendungsmodell (Application Protocol) definiert wurde. Damit entfällt für die Realisierung einer STEP-Produktdatenbank der Datenmodellentwurf für den Datenbankimplementierer. Das Datenmodell der STEP-Produktdatenbank wird in die Entwicklung der Anwendungsprotokolle verlegt. In Kapitel 6 wird die Entwicklung der Anwendungsprotokolle weiter vertieft. Zusammen mit der Realisierung einer genormten Zugriffsschnittstelle auf Datenbestände nach der STEP-Spezifikation wird hiermit eine im Vergleich zu ANSI/SPARC noch weitergehende Entkopplung der Datenmodellierung von der Implementierung eines Softwaresystems erreicht.

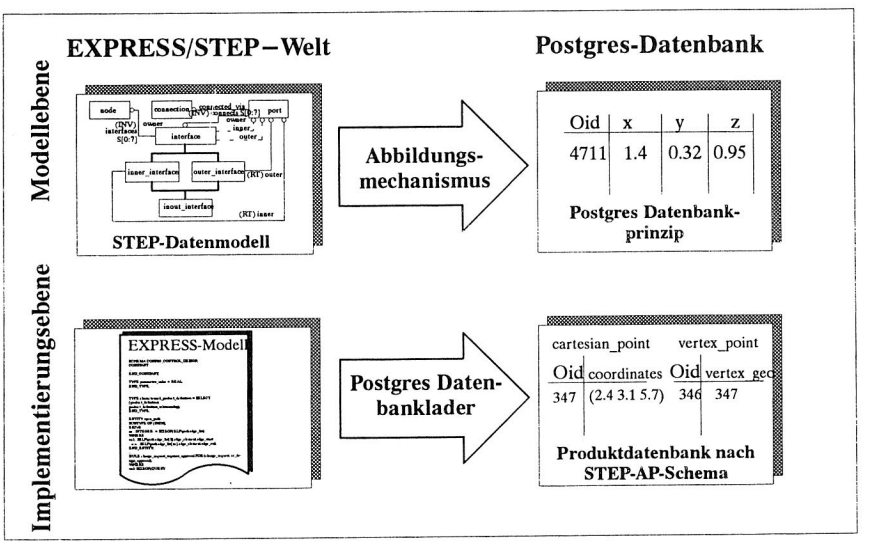


Abb. 24 Trennung von Datenmodellierung und Implementierung für STEP-Datenbanken

Für die Realisierung einer STEP-Produktdatenbank ist das für die Datenbankanwendungssysteme zu unterstützenden STEP-Anwendungsprotokoll das relevante Datenmodell. Für die Realisierung einer STEP-Produktdatenbank aus einem gegebenen EXPRESS-Datenmodell ist aber die grundsätzliche Problematik nicht die direkte Einbringung der in EXPRESS definierten Konstrukte des Datenmodells, sondern die Frage, wie die vorgegebenen EXPRESS-Konstrukte allgemein auf das Datenbankprinzip abgebildet werden können. Dabei ist die Problematik zu verstehen, wie z.B. EXPRESS Typen,

Entities, Regeln, Funktionen, Prozeduren und die dabei unterstützten Prinzipien wie z.B. Vererbung auf Datenbankprinzipien abgebildet werden sollen.

Die Kriterien für eine Abbildungsvorschrift von EXPRESS Konstrukten auf eine Datenbank können etwa wie folgt formuliert werden:

- Effizienz; Die Abbildung von EXPRESS Konstrukten auf das Datenbankmodell soll im Betrieb der Datenbank keine unnötig hohen Performanzverluste aufweisen.
- Strukturelle Unterstützung; Die gewählten Zielkonstrukte der Datenbank sollen möglichst datenbankeigene Prinzipien sein. Dadurch wird oftmals das oben definierte Ziel der Performanzoptimierung erreicht.

Liegt für jedes EXPRESS Konstrukt eine sinnvolle Abbildung auf ein Datenbankkonstrukt vor, so kann für jedes in EXPRESS definierte Datenmodell manuell oder automatisch das Datenbankschema aus dem vorliegenden EXPRESS Schema gebildet werden.

In den folgenden Kapiteln wird diese zweistufige Abbildung für die Implementierung einer STEP-Datenbank für Postgres dargestellt.

4.4.3 Abbildung der EXPRESS-Prinzipien auf das Postgres-Modell

Im folgenden soll die Abbildungsvorschrift von EXPRESS Konstrukten auf das DBMS Postgres erläutert werden. Für jede Abbildung werden die wesentliche Gründe dargestellt sowie Probleme und Einschränkungen diskutiert.

Grundlage der Abbildung auf das Modell von Postgres waren die bekannten Abbildungen auf Relationale Datenbanksysteme. Die erweiterten Möglichkeiten des Postgres Datenmodells wurden konservativ genutzt. Da das Postgres Datenmodell eine Erweiterung des Relationalen Modells darstellt, können alle sinnvollen Abbildungsmechanismen auf das Relationale Modell problemlos übernommen werden. Dort, wo die Abbildung auf dieses Modell an seine Grenzen stößt können die verschiedenen Postgres Erweiterungen genutzt werden.

Die EXPRESS Datentypen lassen sich nach Abb. 6 einteilen. Die Abbildung von EXPRESS Typen in Postgres muß nach zwei Seiten hin unterschieden werden. Zum einen muß die Typisierung von Werten des Laufzeitsystems ermöglicht werden. D.h. es muß ermöglicht werden, daß ein Wert z.B. die Größe 2 als Festkommazahl annehmen kann. Andererseits muß auch ein sogenanntes Metasystem der Typisierung erzeugt werden. Es muß sich also für einen Wert angeben lassen, von welchem Typ dieser Wert ist. In Zusammenhang mit obigem Beispiel muß z.B. unterschieden werden können, ob es sich um einen EXPRESS NUMBER, einen INTEGER-Wert oder aber einen benannten Wert von einem Basistyp INTEGER mit zugeordneten lokalen Regeln (WHERE RULES)

handelt. Während die Auswahl eines Datentyps des Basissystems Postgres die Leistungsfähigkeit des Datenbanksystems bestimmt, wird die Struktur des Meta-Typsystems durch Teil 22 der ISO 10303, der Definition des Data Dictionary bestimmt. Da diese Abbildung vorgeschrieben ist, wird im weiteren hierauf nicht weiter eingegangen.

Für alle Werte von Attributen besteht innerhalb einer Instanzenmenge die Möglichkeit gesetzt oder nicht gesetzt zu sein. Diese Tatsache erfordert die Möglichkeit diesen Sachverhalt innerhalb des Typs oder des Wertes selbst abbilden zu können. Während heute für das C++-SDAI dieser Sachverhalt über Grenzwerte des Attributwertes selbst realisiert wird, wurde hier für jeden realisierten Typ ein Flag eingeführt das es erlaubt, einen gesetzten Wert durch dieses Flag zu erkennen.

Für die numerischen Standarddatentypen Real und Integer bietet sich die direkte Abbildung in die Postgres Datentypen float8 bzw. int4 an. Die EXPRESS Datentypen wie z.B. INTEGER und REAL sind hinsichtlich ihres Wertebereiches und ihrer Genauigkeit für REAL Typen nicht eingeschränkt. Eine Abbildung der EXPRESS Datentypen auf einen Typ einer Programmiersprache, wie dies die Abbildung nach 4-Byte Integer oder 8-Byte Gleitkommazahlen von Postgres bedeutet, ist hiermit eine implementierungstechnische Einschränkung der EXPRESS Typsemantik. Eine Implementierung von EXPRESS wird jedoch mit dem Ziel unternommen, ein Speichermedium von Anwendungssystemen zu sein, die selbst mit der Beschränkung endlicher Wertebereiche zu kämpfen haben. Werden durch die gewählten Abbildung die größten von dem Sprachmodell unterstützten Typen gewählt wird der vom Sprachmodell angebotene Bereich ausgeschöpft. Für ein Anwendungssystem, das sich an eines des Sprachmodelle hält, für die bisher Sprachanbindungen definiert wurden, wird damit keine zusätzliche Einschränkung vorgenommen.

Obwohl in STEP-Austauschdateien auf die Übertragung der Darstellungspräzision von Gleitkommazahlen verzichtet wird, muß diese Informationen prinzipiell in der Datenbank verfügbar sein. Die Abstraktion von diesem Wert würde die Anwendbarkeit von Anwendungssystem u.U. weithin einschränken, so daß diese Information innerhalb des Datentyps abgebildet wird.
Damit läßt sich für Postgres die Struktur des EXPRESS-Real Typs wie folgt abbilden:

```
typedef struct XREAL_ {
char     ind;      /* Gültigkeitsflag */
typename sel;      /* Typidentifikator */
short    prec;     /* Darstellungspräzision */
double   val;      /* tatsächlicher Wert */
} *XREAL;
```

Für Integer gilt entsprechend:

```
typedef struct XINT_ {
char     ind;      /* Gültigkeitsflag */
typename sel;      /* Typidentifikator */
long     val;      /* tatsächlicher Wert */
} *XINT;
```

Der dritte numerische Datentyp NUMBER stellt eine Union der Wertebereiche von Integer und Real dar. Damit muß für den Wertebereich des Datentyps eine Union eines Integerwertes sowie eines Realwertes bereitgestellt werden:

```
typedef struct XNUM_ {
char     ind;      /* Gültigkeitsflag */
typename sel;      /* Typidentifikator */
union {  long      intval
         double realval;}; /* tatsächlicher Wert */
} XNUM;
```

Die logischen Datentypen Boolean und Logical stellen eine zwei- bzw. dreiwertige Logik zur Verfügung. Der Wertebereich des dreiwertigen Typs Logical besitzt die geordneten Werte FALSE < UNKNOWN < TRUE, die des zweiwertigen Typs BOOLEAN FALSE < TRUE. Da für den Typ BOOLEAN die Existenz eines Wertes gesichert ist sowie für den Typ Logical die Nichtexistenz des Wertes explizit angezeigt wird, genügen somit Typ- sowie Wertinformation:

```
typedef struct XLOG_{
typename sel;      /* Typidentifikator */
short    val;      /* tatsächlicher Wert */
} *XLOG, *XBOOL;
```

Die sequentiellen EXPRESS Datentypen STRING und BINARY sind prinzipiell durch den Postgres Datentyp "text" abzubilden. Die prinzipielle Längenunbeschränktheit der EXPRESS Typen wird durch den Postgres Datentyp nicht eingeschränkt. Allerdings kommt hierbei eine systembedingte Begrenztheit von Postgres ins Spiel, wenn die Gesamtgröße eines Tupels betrachtet wird. Diese Größe ist in Postgres auf die Größe einer Speichereinheit begrenzt und beträgt 8kByte. Obwohl Postgres für größere Objekte mehrere sogenannte 'large object interfaces' anbietet, wurde für die Implementierung hierauf verzichtet. Die Performance von Postgres 'large objects' liegt deutlich unter der herkömmlicher Typen. Darüberhinaus werden über 'large objects' keine Operatoren wie Indizierung, Konkatenierung usw. ermöglicht. Vor dem Hintergrund des Ausschlusses von dokumentenartiger textueller Information innerhalb STEP (Textuelle Produktinformation soll nach ISO in der Norm SGML festgelegt werden) wird hiermit keine realisierungstechnische Einschränkung vorgenommen:

```
typedef struct XSTR_ {
char     ind;      /* Gültigkeitsflag */
typename sel;      /* Typidentifikator */
unsigned low;      /* Mindestlänge */
unsigned high;     /* Maximallänge */
string   value;    /* Stringwert */
} *XSTR;
```

Der EXPRESS Aufzähltyp ENUMERATION ist eine geordnete Menge von skalaren Werten, die durch eindeutige Bezeichner repräsentiert werden. Den einzelnen Aufzählungskonstanten werden Integerwerte aufsteigend, von Eins beginnend, zugewiesen. Problematisch bei der Abbildung auf Postgres ist die Tatsache, daß die Aufzählungskon-

stanten trotz numerisch vergleichbaren Wertes nicht kompatibel zueinander sind. Daher ist auch hier die Erhaltung der Information über den Typ der Aufzählungskonstanten nötig.

```
typedef struct XENU_ {
char     ind;      /* Gültigkeitsflag */
typename sel;      /* Typidentifikator */
short    val;      /* Wert */
} *XENU;
```

Für die vier Aggregationstypen ARRAY, BAG, LIST und SET bestehen zwar semantische Unterschiede, die Abbildung in Postgres honoriert diese jedoch nicht strukturell. Vielmehr müssen die Eigenschaften der aggregatwertigen Variablen (z.B. Eindeutigkeit) durch die Operatoren auf den Ausprägungen gesichert werden.

```
typedef struct XARR_ {
char     ind;      /* Gültigkeitsflag */
char     unique;   /* Eindeutigkeitsflag */
typename sel;      /* Typidentifikator */
unsigned lo;       /* unterer Index */
unsigned hi;       /* oberer Index */
typename ct;       /* Komponentenbezeichner */
oid      content[];        /* Elemente */
} *XARR, *XBAG, *XSET, *XLIST;
```

Für den Auswahltyp SELECT muß die Obermenge aller definierbaren Typen abbildbar sein. Insbesondere können Selecttypen selbst wieder Element der SELECT Auswahlmenge sein:

```
typedef struct XSEL_ {
char     ind;      /* Gültigkeitsflag */
typename sel;      /* Typidentifikator */
char     curr_sel;         /* Typ des aktuell eingetragenen Wertes */
union {
         XINT     intval;
         XREAL    realval;
         XLOG     logval;
         XBOOL    boolval;
         XSTR     strval;
         XBIN     binval;
         XENU     enuval;
         XARR     arrval;
         XABG     bagval;
         XSET     setval;
         XLIST    listval;
         XENT     entval;
         XSEL     selval;
} u;
} *XSEL;
```

Die Abbildung des ENTITY-Konstruktes auf Postgres Konstrukte hält sich eng an die Realisierungen, wie sie bei Relationalen Datenbanken vorgenommen wurden (Kapitel 4.2). Für jede EXPRESS Entität wird einer eigene Klasse erzeugt. Die Attribute der EX-

PRESS-Entität werden als Attribute einer Postgres Klasse abgebildet. Die Typen der Entityattribute können als die oben definierten Typen angegeben werden.

```
PQL-Syntax:
create 'Entityname' ('Attributname1' = 'Attributtyp1', ...)
```

Gibt der Typ eines EXPRESS Entityattributes selbst eine Entität an, so wird als Postgres Attributtyp der Postgres Klassenname des Entitytyps des Attributes verwendet. Als Werte dieser Attribute können dann die Oids der referenzierten Tupel angegeben werden. Damit ergibt sich z.B. mit den EXPRESS Definitionen

```
ENTITY mypoint;
x,Y,z : REAL;
END_ENTITY;

ENTITY myedge;
start,end : mypoint;
END_ENTITY;
```

folgende Postgres-Sitzung:

```
create mypoint (x=real, y=real, z=real);
create myedge (start=mypoint, end=mypoint);
append mypoint (x=1, y=2, z=3)
append mypoint (x=2, y=3, z=4)
```

Beim Zugriff auf diese Tupel stellt sich zunächst die Frage, in welcher Klasse dieses Tupel zu finden ist. Wäre der Klassenname nicht bekannt, so müßte jede Klasse daraufhin überprüft werden, ob sie dieses Tupel enthält. Unter dem Gesichtspunkt der Performanz ist diese Vorgehensweise nicht akzeptabel. Über den Typbezeichner eines Attributtyps kann jedoch auf die Klasse, die das Tupel enthält direkt zugegriffen werden. Damit ergibt sich folgende Sitzung mit Postgres:

```
retrieve (mypoint.oid, mypoint.all)
----------------------
| oid    | x | y | z |
----------------------
| 349279 | 1 | 2 | 3 |
----------------------
| 349280 | 2 | 3 | 4 |
----------------------
```

Die (Oid-)Werte der Tupel werden also als Attributwerte in das Tupel *myedge* eingetragen:

```
append myedge (start=349279, end=349280)
retrieve (myedge.oid, myedge.all)

---------------------------
| oid    | start  | end    |
---------------------------
| 349339 | 349279 | 349280 |
---------------------------
```

Die Abfrage der Attributwerte eines entitywertigen Attributes liefert also die Oids der Tupel zurück, die den "Wert" dieses Attributes darstellen. Da die direkte Interaktion mit dem Datenbanksystem hier über interaktive Medien (PQL-Eingabemonitor) erfolgt, wird hier keine automatische Dereferenzierung der Referenzwerte vorgenommen und die wenig aussagekräftigen Oid-Werte ausgegeben. In der Anwendung des Datenbanksystems über Anwendungsprogramme muß diese Dereferenzierung transparent vorgenommen werden. Dies jedoch stellt wie oben erläutert kein prinzipielles Problem dar.

Die durch die EXPRESS Supertyp/Subtyp-Beziehung definierte Vererbungsbeziehung läßt sich zunächst durch die Postquel '*inherits*' Anweisung umsetzen. Für den Fall der Mehrfachvererbung (*multiple inheritance*) bietet Postgres jedoch keine Problemlösung an. Für diesen allerdings seltenen Fall muß ein explizites Prefixing des Attributnamens erfolgen.

Invertierte Attribute können bei der Umsetzung direkt wie explizite Attribute behandelt werden. Dabei ist jedoch zu beachten, daß die Beziehungsrichtung verlorengeht. Die Konsistenzsicherung der durch die inverse Beziehung gestaltete Relation kann in Postgres durch Regeln formuliert werden. Dabei müssen die Fälle des Erzeugens, Veränderns und Löschens der betroffenen Tupel berücksichtigt werden. Darüberhinaus läßt sich ebenfalls die Semantik der Kardinalität der inversen Beziehung sichern.

Abgeleitete Attribute können in Postgres durch das Konzept des Datentyps 'postquel' abgebildet werden. Attribute dieses Typs werden durch eine Berechnungsvorschrift bestimmt die selbst in der DML 'postquel' formuliert werden muß. Werden Werte dieses Typs nachgefragt, so wird die Berechnungsfunktion angestoßen. Auch wenn die Darstellbarkeit von Berechnungen in 'postquel' gewissen Grenzen unterliegt, (fehlende Berechnungsvollständigkeit) ist jedoch über den Aufruf von C-Funktionen aus 'postquel'-Anweisungen heraus jede algorithmisch formulierbare Problemlösung nutzbar.

EXPRESS Regeln können in Postgres mit unterschiedlichem Erfolg realisiert werden: Eindeutigkeitsregeln (UNIQUE) von Attributen einer Instanzenmenge lassen sich über die Postquel Key-Klausel realisieren, solange es sich um nur ein Attribut handelt. Werden hingegen zusammengesetzte Primärschlüssel angegeben kann eine Abbildung in Postgres nicht erfolgen. Dies ist jedoch auf implementierungstechnische Einschränkungen von Postgres zurückzuführen; Kommerzielle Relationale Datenbanken unterstützen dies.

EXPRESS Domain Regeln (WHERE) können durch Postgres Regeln unterstützt werden. Problematisch ist hierbei jedoch, daß im Aktionsteil der Regeln nur elementare Postgres Aktionen auftreten können. Sollte ein Anwendungssystem eine im Sinne des EXPRESS-Modells inkorrekte Anweisung von der Datenbank fordern, so würde die Aktion nicht stattfinden ('instead do nothing'), eine Rückmeldung über diese falsche Operation würde jedoch nicht erfolgen.

4.4.4 Schemagenerator und -Lader für POSTGRES

Der im vorigen Abschnitt dargestellte Abbildungsmechanismus für EXPRESS Modelle auf das Datenbankprinzip von Postgres stellt den Mechanismus der Abbildung bereit. Für die Realisierung einer STEP-Datenbank muß ein gegebenes EXPRESS Modell vollständig in ein Datenbankschema überführt werden. Obwohl eine manuelle Abbildung möglich erscheint, ist durch die Anzahl der in einem sinnvollen EXPRESS Modell vorhandenen Konstrukte[1,2] eine automatisierte Abbildung vorzuziehen. Die Fehleranfälligkeit des Abbildungsvorganges scheint damit geringer und darüberhinaus leichter nachvollziehbarer zu sein.

Auf der Basis des EXPRESS-Parsers expparse wurde ein Schemaübersetzer von EXPRESS-Modellen zu Postgres Datenbankschemata entwickelt. In Abb. 25 wird das Konzept prinzipiell dargestellt. Aus einem vorliegenden EXPRESS Schema wird mit Hilfe des EXPRESS Parsers expparse eine Hauptspeicherstruktur erzeugt, die dann von einem Backend des Parsers in Datenbankschemata umgesetzt wird.

Analog hierzu bietet es sich an, auf Basis des STEP-Datei Parsers p21parse einen Postgres Instanzenlader zu realisieren, der Instanzen von EXPRESS Entitäten in Tupel der Datenbank Postgres überführt.

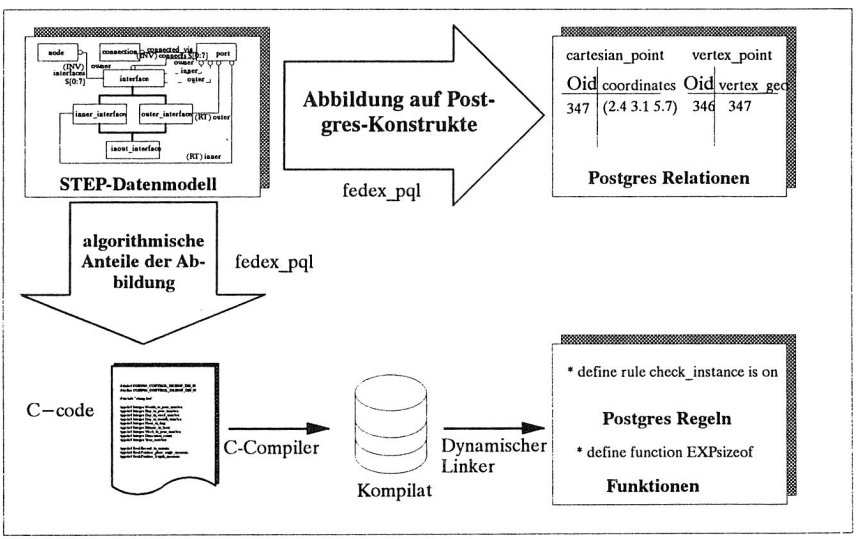


Abb. 25 Abbildung von EXPRESS Modellen durch automatisierte Schemaübersetzung für das DBMS Postgres

Für den Schemaübersetzer bieten sich zunächst zwei unterschiedliche Realisierungsprinzipien an:

12. Das AIM des Teiles 203 der ISO 10303 enthält bereits 249 Entity- und Typdefinition.

Ähnlich wie innerhalb von Schemaübersetzern für Relationale Datenbanksysteme /188/ ist es möglich, Postquel Kommadosequenzen zu erzeugen, die über ein Datenbankinterface der Datenbank zugeführt werden. Problematisch an diesem Prinzip ist die Notwendigkeit, für Einträge in der Datenbank auf die Objektbezeichner (Oids) der Datenbank zugreifen zu können. Da die von Postgres für Objekte vergebenen Bezeichner willkürlich generiert werden, ist es nicht möglich, zur Schemagenerierungszeitpunkt Kenntnis über einen Bezeichner zu besitzen.

Zur Lösung dieses Problems wurde die direkte Kommunikation mit der Datenbank über das Datenbank-API 'libpq' realisiert. Über dieses API können einerseits alle Postquel Kommandos an die Datenbank abgesetzt werden. Die Ergebnisse einer Datenbankanfrage werden als Datensatz zurückgeliefert, der mit bereitgestellten Funktionen traversiert werden kann.

Neben der Erzeugung der Datenbankschemata liefert der Schemaübersetzer die Implementierungen der Ein- Ausgabefunktionen für die in Postgres erzeugten benutzerdefinierten Datentypen der abgebildeten EXPRESS Typen, sowie die innerhalb der EXPRESS Schemata vorhandenen algorithmischen Anteile des Schema. Diese müssen zunächst als C-Code in Dateien erzeugt, compiliert, und über den dynamischen Lader in die Datenbank eingebracht werden (Abb. 25).

Mit der realisierten STEP-Datenbank auf Postgres konnten viele der Einschränkungen, die das relationalen Modell für eben diesen Einsatz problematisch machen gelöst werden. Im Gegensatz zu Datenbankrealisierungen auf objektorientierten Datenbanken bietet Postgres prinzipiell eine für Mehrbenutzerbetrieb geeignete Basis. Die Einschränkungen, mit denen während der Realisierung zu kämpfen waren, sind zumeist auf den prototypischen Realisierungsstand des Softwaresystems zurückzuführen. Bei Nutzung der in jüngster Zeit bekanntgewordenen kommerziellen Variante von Postgres 'Illustra' sollten diese Schwierigkeiten in den Hintergrund treten. Damit wäre eine geeignete Plattform für STEP-Datenbanken auch für den Anwendungsbetrieb möglich.

4.5 Das SDAI als Schnittstelle zu STEP-basierenden Datenbanken

Die Grundlegende Einordnung des SDAI als Zugriffsschnittstelle auf STEP-konforme Daten wurde bereits in Kapitel 2.4.2 vorgenommen. Im Folgenden soll explizit auf die Abbildung EXPRESS-definierter Modelle auf das C++-Sprachmodell eingegangen werden. Die ebenfalls in Entwicklung befindlichen Sprachanbindungen für C und IDL sollen kurz erwähnt werden. Anschließend werden existierende SDAI-Implementierungen verglichen.

Obwohl zunächst anzunehmen wird nicht gefordert, daß über ein SDAI nur auf Datenbestände zugegriffen werden kann, die STEP-konform vorliegen (wobei nur die STEP-Aus-

tauschdatei ein solches Datenformat exakt beschreibt), sondern es wird eine Schnittstelle propagiert die auf ihrer API-Seite Datenbestände so darbietet, als ob sie sich in einem derartigen Format befinden würden. Insbesondere macht dies beliebige Transformationen des Datenbestandes innerhalb dieser Schnittstelle möglich (Abb. 26). Für die Implementierungen von STEP-Datenbanken auf Relationalen Datenbanksystemen müssen hier aufgrund des Typ- oder besser Prinzipmismatches derartige Transformationen in größerem Umfange vorgenommen werden als dies für Objektorientierte Datenbanken der Fall ist.

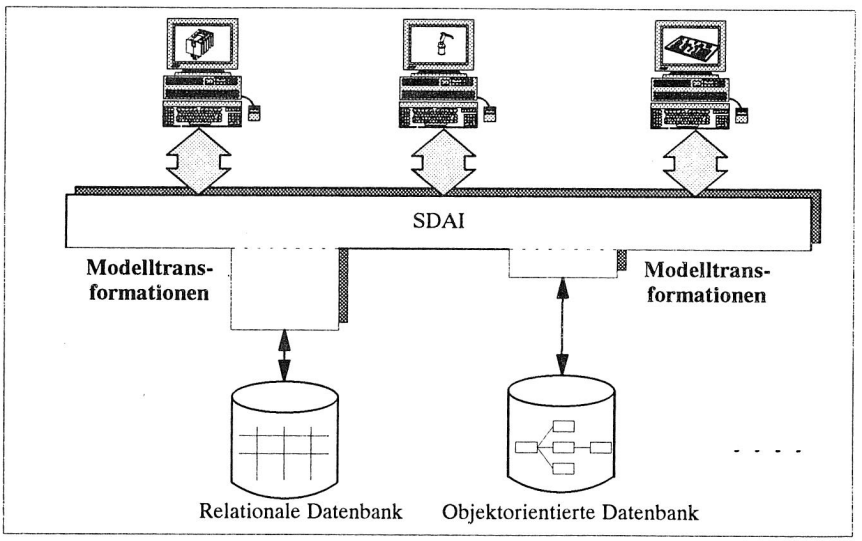


Abb. 26 SDAI - Einsatz zur Verwaltung von Datensätzen im STEP-Format

Während die C++-Sprachanbindung des SDAI also sowohl die Abbildung des EXPRESS-Sprachmodells auf C++ als auch die Bildung der Zugriffsschnittstelle für C++-Anwendungssysteme darstellt[13], muß die Abbildung von EXPRESS auf relationale Datenbanken sowie die Realisierung eines SDAI auf relationalen Datenbanken als ein zweistufiger Prozeß verstanden werden.

4.5.1 Die C++-Sprachanbindung des SDAI

Die Definition der C++-Sprachanbindung definiert in ihrer Methodik eine Abbildung von EXPRESS-Modellen auf das C++-Sprachmodell. Sie ist damit einerseits der Mechanismus um EXPRESS-Modelle auf objektorientierte Datenbanken abzubilden, wenn diese das C++-Sprachmodell persistent implementieren.

---

13. Dies gilt zumindest für objektorientierte Datenbanken, die auf dem C++-Sprachmodell basieren.

Prinzipiell macht sich die C++-Sprachanbindung stark von dem Prinzip der Abstraktion über Klassen von C++ Gebrauch. Außer für die einfachen Datentypen Integer und Real werden keine Klassen benötigt.
So wird der EXPRESS Datentyp INTEGER als ein Typ interpretiert, der durch einen 32-bittigen Wert repräsentiert werden kann. Für den EXPRESS Typ REAL wird eine Fließkommadarstellung mit doppelter Genauigkeit nach IEEE gefordert. Dies wird zumeist durch eine 8 Byte Fließkommazahl repräsentiert werden können. Interessanterweise wird in der SDAI-Spezifikation der EXPRESS-Datentyp NUMBER derart behandelt, als würde es sich um einen Auswahltyp (SELECT) zwischen INTEGER und REAL handeln. Dies stützt die in Kapitel 2.3.4 vertretende These, der EXPRESS Datentyp NUMBER sei kein in EXPRESS notwendiger Basistyp.

Für die logischen Datentypen LOGICAL und BOOLEAN werden C++-Aufzählungstypen und Klassen wie folgt definiert /45/:

```
enum Logical { LFalse, LTrue, LUnset, LUnknown };

class LOGICAL {
public:
        LOGICAL();
        LOGICAL (Logical state)
        ...
}
```

Während mit dem Auswahltyp der Wertebereich von Variablen dieses Typs festgelegt werden, erlauben die Klassendefinitionen das Verhalten von Operationen auf Variablen diesen Typs (hier etwa Gleichheit, Ungleichheit, Typkonvertierungen usw.) zu steuern. Die C++-Klasse LOGICAL muß in der Lage sein, einen Wert des Aufzähltyps Logical darzustellen. Eine Vorschrift für eine Membervariable besteht jedoch nicht. Für den Datentyp BOOLEAN wird eine analoge Abbildung vorgenommen.

Der EXPRESS Datentyp STRING wird (derzeit) durch die beiden Klassen String_var und String repräsentiert. Die Klasse String_var enthält eine durch ein Nullzeichen abgeschlossene Zeichenkette. Weiterhin sind für diese Klasse Konstruktoren, ein Destruktor sowie einige Zugriffsoperatoren definiert. Für die von String_var abgeleitete Klasse String soll das Verhalten des EXPRESS Typs STRING implementieren.

Für die Abbildung des EXPRESS Datentyps BINARY wird auf eine derartige Klassenhierarchie verzichtet. Die Klasse Binary implementiert Wert und Verhalten des Typs; eine Vorschrift für die Abbildung des Wertes analog zu der Zeichenkette in der Klasse String wird für Binary nicht vorgeschrieben.

Aggregatwertige Typen werden durch Klassen repräsentiert. Für die Namensgebung benannter Typen wird die Namensgebung durch Konkatenation des Elementnamens, zweier Unterstrichzeichen ("_") sowie dem Namen des Aggregattyps. Das erste Zeichen des Elementnamens soll groß, alle anderen Zeichen klein geschrieben werden. Aus der EXPRESS Definition:

```
LIST OF point
```

wird also die C++ Definition:

```
class Point__list { ..
```

Diese Klasse stellt die Schnittstelle zu einer Variablen dieses Typs dar. Es werden Methoden zum Erzeugen, Vernichten und tiefen Kopieren (*deep copy*) definiert. Die Implementierung des Verhaltens wird in einer Klasse realisiert, deren Klassennamen durch Konkatenation aus dem Elementnamen, zwei Unterstrichzeichen, der Zeichenkette "sdai" und dem Namen des dem Aggregate zugrundeliegenden Elementes. Der Anfangsbuchstabe des Klassennamens ist groß, alle anderen Zeichen klein. Aus obigem Beispiel wird dann:

```
class Point__sdailist { ...
```

Innerhalb der zweiten Klasse soll das Verhalten einer Variablen des Aggregattyps realisiert werden. Innerhalb dieser Klasse werden die für das SDAI definierten Funktionen als C++ Methoden realisiert.

EXPRESS Aufzählltypen werden durch einen C++ Aufzählltyp sowie durch eine Klasse repräsentiert, die einen Wert vom Typ des Aufzählltyps trägt. Der Namen des Aufzählltyps entspricht dem EXPRESS Typnamen mit großgeschriebenen Anfangsbuchstaben, die Elemente des Aufzählltyps erhalten als Präfix den Namen des Typs mit zwei Unterstrichzeichen, die Klasse erhält stattdessen das Suffix "_var". Die Elemente des Aufzählltyps werden in der Reihenfolge der Definition innerhalb des EXPRESS-Modells angegeben; der Initialwert trägt den Integerwert 0.

Für EXPRESS Auswahltypen existieren für die zwei Konformitätsklassen unterschiedliche Implementierungsvorschriften: Für die Konformitätsniveau 1 wird eine diskriminierende Union der Elemente des Auswahltyps sowie als Diskriminator ein Aufzählltyp der Elemente. Der Aufzählltyp trägt zusätzlich zum Namen des EXPRESS Auswahltyps den Suffix "_sdaiselect", die Elemente des Aufzählltyps den Suffix des Elementnamens, die Klasse den Namen des EXPRESS Auswahltyps. Damit ergeben sich z.B. aus dem EXPRESS SELECT Typ:

```
TYPE arbitrary_choice = SELECT (STRING, REAL, point); END_TYPE
```

die C++-Konstrukte:

```
enum        Arbitrary_choice__sdaiselect        {Arbitrary_choice__String,
Arbitrary_choice__Real, Arbitrary_choice__Point};

class Arbitrary_choice {
...
}
```

Für die Ausprägungen jedes Elementes werden dann Zugriffs- und Setzfunktionen definiert. Die damit bereitgestellte Schnittstelle ist äußerst barock und entspricht wenig dem

objektorientierten Charakter einer C++-Sprachanbindung.
Die Abbildungsvorschrift für Konformitätsniveau 2 erfüllt diesen Anspruch besser: Es wird eine Basisklasse Select definiert, die gemeinsame Funktionalität aller Select-Klassen bereitstellt. Für jeden EXPRESS SELECT Typ wird eine Klasse vereinbart, die als Präfix des EXPRESS Typnamens 'Select_' trägt. Es sind Konstruktoren aus den zugrundeliegenden Elementen sowie Konversionsfunktionen zwischen den Elementen definiert:

```
TYPE person : SELECT (employee, student); END_TYPE;

ENTITY employee; ssn : STRING; END_ENTITY;

ENTITY student; id : INTEGER; END_ENTITY;

class Select_Person : public Select {
public:
        Select_Person (Employee_ptr&);      // Konversionskonstruktor
        Select_Person (Student_ptr&);       // Konversionskonstruktor
        operator Employee_ptr();            // Konversionsoperator
        operator Student_ptr();             // Konversionsoperator
        String_var Ssn();
        void Ssn(const String_var&);
        Integer Id();
        void Id(Integer);
        Select_Person& operator=(Employee_ptr&);
        Select_Person& operator=(Student_ptr&);
        Logical _Student() const;
        Logical _Employee() const;
        String_var Underlying_typeName() const;
#ifdef SDAI_CPP_LATE_BINDING
        const Named_type_Ptr Underlying_type() const;
#endif
        Select_Person();                    // Standardkonstruktor
}
```

EXPRESS benannte Typen (*defined type*) werden auf C++ Typdefinitionen umgesetzt. Die bei der Definition möglich Angabe von Restriktionen (WHERE Rules) werden nicht durch Restriktionen der C++-Typisierung abgebildet, sondern es wird nur verlangt, daß die Validierung von Attributen von Anwendungsinstanzen, deren Typ eine Restriktion besitzt, explizit durch Funktionsaufruf ('ValidateWhere_rule()') erfolgen kann.

Die Abbildung von EXPRESS Entitäten erfolgt nach C++ Klassen. Der Zugriff auf Attribute von Entitäten wird durch Zugriffsfunktionen der jeweiligen C++-Klasse ermöglicht. Die Typisierung der Setz- und Abfragemethoden wird durch den Typ des EXPRESS Attributes (bzw. dessen abgebildetes C++-Äquivalent) gegeben. Die Definition der EXPRESS Attributwerte als private Elementvariablen wird damit nicht vorgeschrieben, ist aber implizit gegeben.
Für Attribute, deren Namensbildung durch Mehrfachvererbung nicht eindeutig gestaltet werden kann wird vorgeschlagen, in der Basisklasse den Klassennamen als Attributpräfix zu verwenden. Hier wird auf die Fähigkeit von C++ verzichtet, diese Mehrdeutigkeiten durch den Namensscope-Operator '::' aufzulösen:

```
ENTITY a;
        attr : INTEGER;
END_ENTITY;

ENTITY b;
        attr : REAL;
END_ENTITY;

ENTITY c
SUBTYPE OF (a, b);
END_ENTITY;

typedef class A* A_ptr;
typedef A_ptr A_var;

class A {
...
Real A_Attr(void) const;
void A_Attr(Real);
};

class B {
...
Integer B_Attr(void) const;
void B_Attr(Integer);
}

class C : public A, public B {}
```

Die durch die in inversen Attribute modellierte Beziehung zwischen Entitäten wird in der C++-Anbindung explizit realisiert. D.h. die Assoziation wird beiderseits durch explizite Attribute ausgedrückt. Die Richtung der Assoziation geht damit in der C++-Anbindung verloren.

Eines der wesentlichsten Probleme der Abbildung der EXPRESS Entitäten nach C++-Klassen stellt sicherlich die Bereitstellung von C++Klassen für die mittels EXPRESS implizit definierten komplexen Entityinstanzen dar. Da eine C++-Variable die Instanz genau einer Klasse darstellt, muß für jede innerhalb eines EXPRESS-Schemas definierte komplexe Entityinstanz eine weitere C++Klasse bereitgestellt werden. In /45/ wird die explizite Definition der möglichen komplexen Entityinstanzen durch eine Erweiterung der EXPRESS-Syntax gefordert. Diese Forderung ignoriert jedoch die Problematik der Bestimmung dieser Menge. Wie in /13/ und /162/ gezeigt ist diese Menge zwar berechenbar, jedoch ist die Berechnungszeit selbst für relativ einfache Schemata wie dem aus /61/ verfügbaren, bisher nicht akzeptabel. Ein möglicher Ausweg aus diesem Dilemma scheint nur durch die Definition von komplexen Entityinstanzen zur Laufzeit durch die Funktion 'GetComplexEntity' möglich. Damit erscheint ein Mischbetrieb des SDAI zwischen early und late-Anbindung zwingend nötig.

Die Abbildung abgeleiteter Attribute (DERIVE) wird derart vorgenommen, daß alle abgeleiteten Attribute eines Schemas als eine Funktion einer Klasse (Model_contents_>>Schemaname<<) realisiert werden. Damit ist ein abgeleitetes Attribut nicht

innerhalb der Klasse verfügbar, sondern muß über einen Umweg angesprochen werden. Die Realisierung als Zugriffsfunktion innerhalb der dem Entity zugeordneten Klasse wäre hierfür eine wesentlich sinnvollere Lösung.

Insgesamt macht der aktuell vorliegende Normvorschlag einen weitgehend unausgereiften Vorschlag. Ob die hier identifizierten Probleme bis zur Entscheidung der Norm noch ausgeräumt werden können scheint fraglich. Hingegen dürfte die Qualität dieses Teils der Norm einen erheblichen Einfluß auf die Akzeptanz von Anbietern haben und damit die weitgesteckten Ziele von STEP beeinflussen.

4.5.2 Andere Sprachanbindungen

Neben der in Kapitel 4.5.1 analysierten Sprachanbindung des SDAI existieren darüberhinaus weitere Sprachanbindungen für C (/46/) sowie für IDL /47/. Die C Sprachanbindung an C stellt eine Anbindung nach dem Prinzip des späten Bindens (late-binding) dar, d.h. die Implementierung der C-Anbindung ist unabhängig von einem gegeben EXPRESS-Schema. Der Unterschied zwischen der frühbindenden (early-binding) C++Sprachanbindung und der spätbindenen (late-binding) C-Sprachanbindung liegt also konzeptionell in der Attributmanipulation von Objekten. Attributmanipulation über spätgebundene Funktionen erfolgen durch Identifikation der Instanz sowie des Attributes innerhalb der Parameterliste der Manipulationsfunktionen. Mit der Entitydefinition:

```
ENTITY point;
        x : REAL;
        y : REAL;
        z : OPTIONAL REAL;
END_ENTITY;
```

folgt mit der Zugriffsfunktion:

```
void*     sdaiGetAttr(SdaiInstance     instance,     SdaiAttr     attribute,
SdaiPrimitiveType valueType, void *value);
```

der Aufruf:

```
SdaiInstance my_point;
SdaiAttr x_attr;
SdaiPrimitiveType valuetype
void* value;
...
sdaiGetAttr(myinstance, x_attr, valuetype, value);    ???<-Bockmist
```

Offensichtlich ist dieser Version einer frühgebundenen hinsichtlich Typsicherheit und Performanz deutlich unterlegen. Die Überprüfung der Parametertypen muß zur Laufzeit erfolgen. Hingegen ist eine Implementierung eines spätgebundenen SDAI-Sprachanbindung wesentlich flexibler hinsichtlich der unterstützten EXPRESS-Modelle und damit für beliebige Anwendungsprotokolle nutzbar.

In der in jüngster Zeit aufgenommenen Entwicklung einer IDL (*Interface Definition Language*) Sprachanbindung wird versucht, die Implementierung einer netzwerkweit verteilten Zugriffsfunktionalität mittels CORBA zu ermöglichen. Offenbar wird dabei auch versucht, die Brücke zu der sich abzeichnenden Akzeptanz der Modellierungssprache IDL der OMG zu erreichen. Hinsichtlich Unterstützung weiterer Funktionalität zur Datenverwaltung wird damit nicht erreicht; die Verteilung von Datenbeständen wird durch Datenbankfunktionalität bereits transparent unterstützt.

Die Entwicklung der FORTRAN Sprachanbindung wurde kürzlich eingestellt. Für die ehemals im technisch-wissenschaftlichen Bereich vorherrschende Sprache besteht offensichtlich kein Bedarf mehr. Vor allem durch Neuentwicklungen im CAD-Bereich scheint hierfür die Nachfrage verlorengegangen zu sein.

4.5.3 Analyse von SDAI-Implementierungen auf Datenbanksystemen

Obwohl die SDAI-Entwicklung sowie die Sprachanbindungen zur Zeit noch keine stabile Spezifikationen hervorgebracht haben, existieren bereits einige Realisierungen sowohl im wissenschaftlichen als auch im kommerziellen Bereich.

Im Rahmen des ESPRIT-Projektes ProDEX wurde ein C-Binding SDAI realisiert, das den Zugriff auf Datensätze innerhalb lokaler Dateien ermöglicht. Ziel dieser Entwicklung war nicht die Unterstützung des konkurrierenden Zugriffs auf eine Datenbasis, sondern die modulare Vereinheitlichung der Entwicklung von Prozessoren für CAD/CAE-Systeme. In Ermangelung der Spezifikation eines API für Dateizugriffe mag dies sicherlich gerechtfertigt sein. Eine Lösung für den allgemeinen Zugriff auf das Modell bzw die Produktdatenmenge stellt dies allerdings wegen der Abstraktion des SDAI von algorithmischen Anteilen eines EXPRESS-Modells nicht dar.

Aus der Entwicklung einer an EXPRESS orientierten C++-Klassenbibliothek (ROSE) am RPI (Rensslaer Politechnic Institute) gingen C++-Code-Generatoren für early-binding SDAI sowie Adapter für die objektorientierten Datenbanksysteme Objectstore, Versant und OODB sowie das Relationale Datenbanksystem Oracle. Aufgrund der sehr frühen Realisierungsphase des SDAI nach einem CD-Dokument muß das SDAI heute als nicht mehr SDAI-konform bezeichnet werden. Ob die Implementierung an die heute gültige Version der Spezifikation angepaßt wird oder der Normstatus des SDAI abgewartet wird, ist nicht bekannt. Die Realisierung als solches muß insgesamt wohl als Forschungsprototyp bezeichnet werden.

Von der Firma GOPAS wird auf für ONTOS ein Werkzeugsatz entwickelt, der u.a. ein C++-SDAI enthält. Besonderheit dieser Entwicklung ist die Tatsache daß der Codegenerator wahlweise SDAI-konformen aber offensichtlich wenig performanten oder aber nichtkonformen 'lean' Code erzeugt. Letzterer zielt allerdings auf die manuelle Anpassung durch einen Anwendungsprogrammierer. Ob diese Vorgehensweise hinsichtlich der Komplexität des erzeugten Codes realistisch ist, bleibt fraglich.

In /186/ wird die Abbildung von EXPRESS-Modellen auf C++-Klassen unter Berücksichtigung der Spracherweiterungen von ObjectStore vorgenommen. Die Automatische Übersetzung von EXPRESS-Modellen in Datenbankmodelle wird durch eine Neuimplementierung eines C++-Klassengenerators nach dem EXPRESS 'Pretty Printer' aus dem NIST-Toolkit vorgenommen. Ergebnis dieser Entwicklungen war die Analyse der Realisierbarkeit des SDAI-Vorschlages /45/. Dabei werden einige Problembereiche Identifiziert sowie Verbesserungen Vorgeschlagen. Diese Implementierung dürfte den aktuellen Vorschlag bestmöglichst realisieren.

Die Entwicklung in /3/ muß in diesem Zusammenhang ebenfalls hier genannt werden. In dieser Arbeit wird zwar kein SDAI realisiert, sondern eine Abbildung von EXPRESS nach C++ vorgenommen sowie eine Abbildung von EXPRESS in eine Relationale Datenbank, jedoch wird durch die Realisierung einer C++-Bibliothek aus der EXPRESS-Beschreibung die Definition der C++-Sprachanbindung des SDAI vorweggenommen. Trotz fehlender SDAI-Orientierung wird die Problematik des Datenzugriffs auf die Daten der Datenbank erkannt und ein Check-Out/Check-In-Mechanismus vorgeschlagen.

5 STEP-Implementierungen zur geometrischen Produktrepräsentation

STEP-Implementierungen beziehen sich auf Anwendungssysteme, die einen Anwendungsbereich im Bereich der Produktdatenbeschreibung bzw. -verarbeitung definieren. Die grundlegenden Anwendungen, die zur Beschreibung von Produkten sind CAD-Systeme. Im folgenden wird auf die Implementierung von STEP-kompatiblen Anwendungen im Umfeld der rechnerunterstützten Konstruktion Bezug genommen.

Zunächst ist das Verständnis für die Basis von CAD-Systemen, die geometrischen Modelliersysteme bereitzustellen. Weiterhin werden dann die beiden Implementierungsformen für STEP, die STEP-Austauschdatei und das SDAI in Ihrer Bedeutung für CAD-Systeme anhand von Implementierungsbeispielen erklärt.

5.1 Prinzipien geometrischer Teilemodelle

CAD-Systeme sind durch zwei wesentliche Eigenschaften charakterisiert:

- Die Charakteristik des internen Geometriemodells
- Die Funktionalität, die die Manipulation des internen Modells ermöglicht.

Da das interne Modell die Funktionalität zur Manipulation wesentlich definiert, kommt ihm die grundlegende Bedeutung für ein CAD-System zu. Die Klassifikation von Modellen zur Beschreibung realer technischer Objekte kann durch unterschiedlichen Kriterien vorgenommen werden. Der Begriff des technischen Objekts soll für die folgenden Ausführungen unter der Einschränkung eines Körpers im festen Aggregatzustand in einem dynamischen Gleichgewicht gelten. Nichtfeste Körper sowie Körper, die sich nicht in einem dynamischen Gleichgewicht befinden sollen hierdurch der Betrachtung ausgeschlossen werden.
Mögliche Klassifikationskriterien können sein (Abb. 27):

- die geometrische Dimensionalität der Objektrepräsentation. Die Unterscheidung zwischen zweidimensionalen zweieinhalbdimensionalen oder dreidimensionale Modelle ist eine der häufigst vorgenommenen Kriterien. Mit einer hinreichen hohen Durchsetzung von dreidimensionalen Gesamtmodellen wird dieses Kriterium seine Bedeutung verlieren.
- die Kardinalität zwischen realem Objekt und dessen Darstellbarkeit in dem zugrundeliegenden Modell. Die häufig geforderte Eindeutigkeit einer Modellbeschreibung kann durch viele Modelle nicht erreicht werden. Zweidimensionale Modelle sind grundsätzlich hierunter einzuordnen. Bei der Betrachtung der Kardinalität zwischen realem Objekt und modellhafter Objektbeschreibung sind n:1 und 1:n Kardinalitäten zu unterscheiden. Linienmodelle (auch dreidimensionale) sind prinzipiell erstgenannter Klasse (für ein Linienmodell lassen sich mehrere reale Objekte finden),

CSG-Modelle (*Constructive Solid Geometry*) sind letztgenannter Klasse zuzuordnen (für ein technisches Objekt lassen sich mehrere Modellbeschreibungen angeben). Durch logische Betrachtung des geschilderten Sachverhaltes kann geschlossen werden, daß n:1 Kardinalitäten für eine modellhafte Beschreibung eines Sachverhaltes nicht adäquat ist; damit läßt sich folgern, daß Linien- oder Drahtmodelle keine geeigneten Modellrepräsentation technischer Objekte sein können.

- die Exaktheit der Darstellung eines Modells. Läßt sich für die Sollgestalt eines Objekt immer eine exakte Darstellung angeben, handelt es sich um ein exaktes, anderenfalls um ein sogenanntes approximatives Modell. Mit Hilfe von CSG-Modellen oder Boundary-Representation Modellen (B-Rep) lassen sich Objekte prinzipiell exakt beschreiben. Die Realität der Begrenztheit der unterstützten Flächentypen in einem B-Rep oder CSG-Modell läßt jedoch auch diese Modelltypen durch die Anwendung von Parameterflächen zu Approximativmodellen werden. Polyedermodelle (*Polyhedras*) sind Modelle nach dem mathematischen Prinzip der r-Mengen (Varianten hiervon sind Spatial Decomposition oder Octtrees) sind dagegen prinzipiell Approximativmodelle. Die Tatsache, daß Objekte existieren, die auch mit diesen Modellen exakt beschrieben werden können, heilt nicht deren modellinterne Unzulänglichkeit.

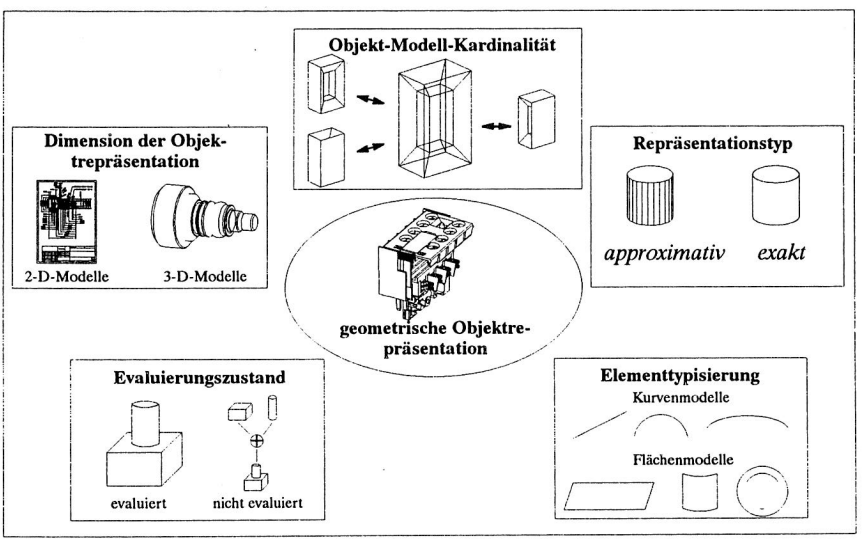


Abb. 27 Klassifikationskriterien für rechnerinterne geometrische Teilemodelle

- sowie die Beziehung eines Modells mit seiner externen Darstellung. Hierbei sind sogenannte evaluierte oder nicht evaluierte Modelle zu unterscheiden. Zu den klassischen Vertretern der ersten Kategorie zählen die B-Rep-Modelle, die CSG-Modelle hingegen letztgenannter Kategorie. Auch die oftmals aus zweidimensionalen Modelle hervorgegangenen zweieinhalbdimensionalen Modelle sind durch eine nicht

evaluierte Beschreibung darstellbar. Die zuletzt in die Geometrieverarbeitung eingegangenen parametrischen Modelle lassen sich den nicht evaluierten Modellen zurechnen.

- Unterstützung der modellinternen Elementtypen und deren Struktur. Bei den B-Rep-Modellen z.B. läßt sich eine Unterscheidung nach den im Modell verfügbaren Geometrieelemente vornehmen. Dabei sind Linien oder Flächenmodelle zu unterscheiden. Durch zusätzliche Aufnahme von Strukturinformation in das Modell, die durch die Topologie der geometrischen Elemente ausgedrückt wird, kann darüberhinaus eine quasivoluminöse Beschreibung eines technischen Objektes erreicht werden.

Die Technische Zeichnung, die in der Technischen Auftragsabwicklung das Fundament der Produktbeschreibung darstellt hat in der rechnergestützten Datenverarbeitung ihre Repräsentation in zweidimensionalen zumeist linienorientierten 'Zeichnungsmodellen' gefunden. Mit einer hinreichen großen Anzahl von Technischen Zeichnungen wird versucht ein technisches Objekt durch Erstellung seiner Ansichten zu beschreiben. Obwohl die Technische Zeichnung eine industriell hohe Bedeutung besitzt wird im weiteren nicht weiter auf diese Modellbeschreibung eingegangen. Die wissenschaftlich eher einfache Beschreibung einer Technischen Zeichnung ist für das weitere Verständnis von geringer Bedeutung, darüberhinaus sind die geometrischen Elemente, die in einer Technischen Zeichnung zur Modelldarstellung benutzt werden sämtlich eine Teilmenge der Geometrieelemente, die in B-Rep-Modellen benutzt werden können. Somit läßt sich deren Behandlung unter der der B-Reps subsumieren.

Nicht strukturunterstützte (topologiefreie) Flächenmodelle finden gerade im Werkzeugbau weitverbreitete Anwendung. Jedoch auch die Behandlung dieser Modelle soll nur unter der Einbeziehung in B-Rep-Modelle stattfinden, zu er sie eine echte Teilmenge darstellen.

Die für die Realisierung von STEP-Implementierungen zwei wesentlichen geometrischen Modellschemata die zur Beschreibung vor allem von Festkörpermodellen, die 'Boundary-Representation' Modelle sowie die CSG-Modelle sollen im Folgenden näher betrachtet werden.

5.1.1 Modelle mit Begrenzungselementdarstellung

Begrenzungselementmodelle (Randdarstellungen, *Boundary Representation Model*, B-Reps) sind Modelle, die ein technisches Objekt durch geometrische Elemente sowie durch Relationen dieser geometrischen Elemente, die durch topologische Elemente angegeben werden, in ein komplexes Modellnetzwerk beschreiben. Das topologische Teilmodell stellt ein Skelett des Modells dar, in dem die geometrischen Nachbarschaftsbeziehungen der geometrischen Elemente repräsentiert werden. B-Rep-Modelle werden in vielen Variationen zur Darstellung technischer Objekte genutzt. Darunter sind sowohl zwei- und dreidimensionale Linienmodelle als auch dreidimensionale Linien- Flächen- und Volumenmodelle zu finden.

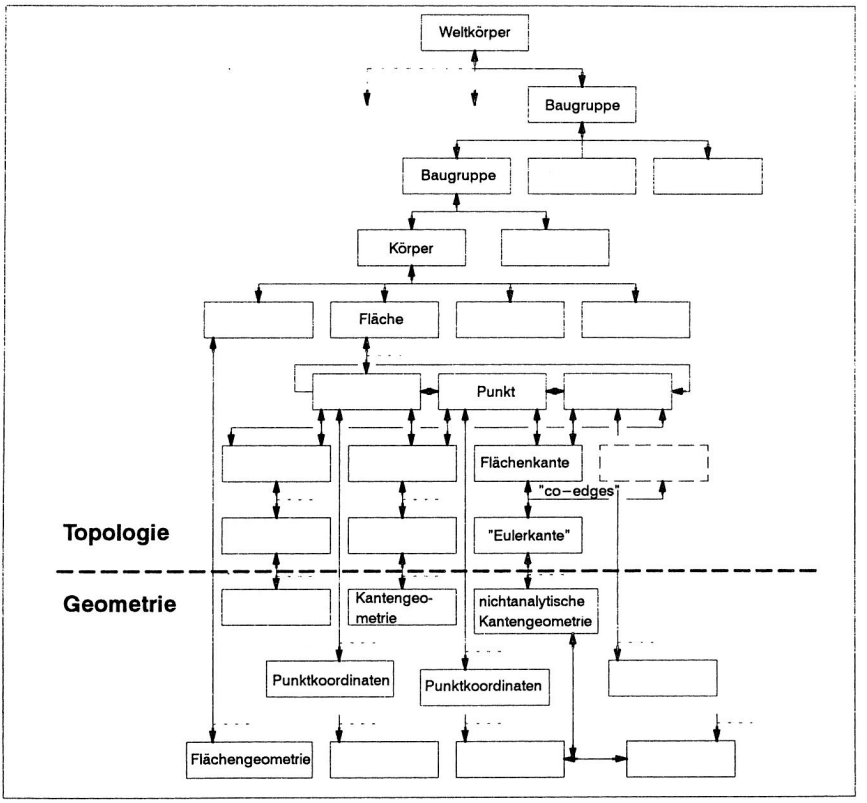


Abb. 28 Aufbau von eines B-Rep-Modelles (nach /140/)

B-Rep-Volumenmodelle stellen die am weitest verbreiteten Volumenmodelle dar /176/. Bei ihnen wird das zu beschreibende Volumen durch eine Menge von topologisch verbundenen Flächen beschrieben, die die Oberflächen des technischen Körpers darstellen. Mehrere solcher Flächenverbünde können dabei mehrere Schalen (*Shells*) des Körpers darstellen, wie dies bei Hohlräumen des Körpers nötig ist. Die Flächen des Körpers werden durch Kantenzüge begrenzt. Die Kantenzüge bestehen aus einer Menge von Kanten, die die zunächst unendlich ausgedehnten geometrischen Flächen nach innen und außen begrenzen. Die einzelnen Kanten des Kantenzuges werden wiederum durch Scheitelpunkte (*Vertex*) begrenzt. Die heute bekannten B-Rep-Modelle unterscheiden sich in Nuancen dieses topologischen Modells sowie in der Art der geometrischen Elemente die die Gestalt des Körpers bestimmen.

Die die Charakteristik eines B-Rep-Geometriemodells wesentlich definierenden Elemente sind die durch das Modell unterstützten Flächentypen. Weit verbreitet sind die analytischen Typen ebene, zylindrische, konische, kugelige und torodiale Fläche. Dar-

überhinaus unterstützen die meisten Modelle auch nichtanalytische, als parametrisierte Flächen dargestellte Flächentypen.

In Abb. 28 wird der allgemeine Aufbau eines B-Rep-Modells dargestellt. Wesentliches Kennzeichen ist die weitgehende Trennung des topologischen sowie des geometrischen Teilmodells. Dies ist ein wesentliches Kriterium für die algorithmisierte Verarbeitbarkeit solcher Modelle. Wünschenswert ist in diesem Zusammenhang eine weitgehende Unabhängigkeit der topologischen Ausprägung eines Modells von der geometrischen Charakterisierung. Als Beispiel hierfür ist die Existenz zweier Endpunkte eines Kante genannt, die auch dann gegeben sein sollte, wenn es sich bei der Kante um einen geschlossenen Kreis handelt.

B-Rep-Modelle existieren in vielen unterschiedlichen Ausprägungen. Einen interessanten Spezialfall der B-Rep-Modelle stellen die sogenannten *Polyhedras* dar, die als Flächentypen einzig die Ebene, und als Kurventypen einzig die gerade Linie unterstützen. Derartige Modelle zeichnen sich durch einfache interne Algorithmen und damit hoher numerischer Stabilität aus. Nachteilig ist allerdings das Prinzips des Approximationsmodells sowie die Abhängigkeit des Antwortzeitverhaltens von der gewählten Approximationsgenauigkeit.

Die meisten B-Rep-Modelle lassen als Modellierungsobjekte nur sogenannte 2-mannigfaltige (non-manifold) Objekte zu. Die Definition besagt daß ein Punkt auf einer 2-Mannigfaltigkeit eine beliebig kleine Umgebung besitzt, die topologisch einem Kreis in der Ebene entspricht (siehe Abb. 29). Die Orientierung auf solche Objekte liegt in der Unterstützung des Beschreibungsvorganges technischer Objekte begründet. Nicht 2-mannigfaltige Objekte stellen Dinge dar, die in der technischen Realität nicht erzeugbar sind.

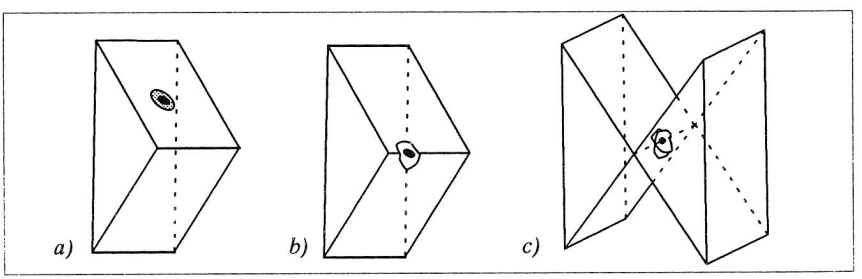


Abb. 29 Objekte, die 2-Mannigfaltigkeiten darstellen (a, b) sowie eine Nicht-2-Mannigfaltigkeit (nach /176/)

B-Rep-Modelle haben ihre Bedeutung durch die Nähe ihres Modells zu den grafischen Darstellungsmodellen erhalten. Sie besitzen Vorteile in interaktiven Anwendungssystemen, die dem Anwender ein graphisches Abbild des Modellierungsobjektes anbieten müssen. Darüberhinaus bieten sie Vorteile bei Anwendungen, die das Modellierungsobjekt geometrisch-topologisch analysieren müssen.

5.1.2 Das Prinzip der Standardvolumenelemente

Das zweite wesentliche Prinzip zur Darstellung von Volumenkörpern stellen die sogenannten 'Constructive Solid Geometry' (CSG-Modelle) dar. CSG-Modelle erzeugen über elementare Verknüpfungsoperationen, üblich sind Durchschnitt, Vereinigung und Schnitt. Mit Hilfe dieser Operationen und verfügbaren Raumprimitiven, hier sind Würfel, Quader, Zylinder, Kugel, Torus u.a. gegeben, wird ein technisches Objekt als ein Baum von Operationen (Knoten) und Operanden (Blätter) als Primitive oder Teilbäume beschrieben (siehe Abb. 30). Zusätzlich lassen sich auch räumliche Transformationen als Operationen angeben.

Die Modellierung mit Hilfe von CSG-Modellen ist stark von der Anwendung der sog. regularisierter boolescher Operationen gekennzeichnet, die dann direkt ein Teil des Modells werden. Da diese Operationen aber nicht kommutativ sind, muß eine Ordnung des CSG-Baumes vorliegen. (Eine Konstruktionshistorie, wie oftmals vermutet wird, beinhaltet ein CSG-Baum also nicht. Ebenso ist Verfügbarkeit von regularisierten Boolschen Operationen kein Hinweis auf ein CSG-Modell.)

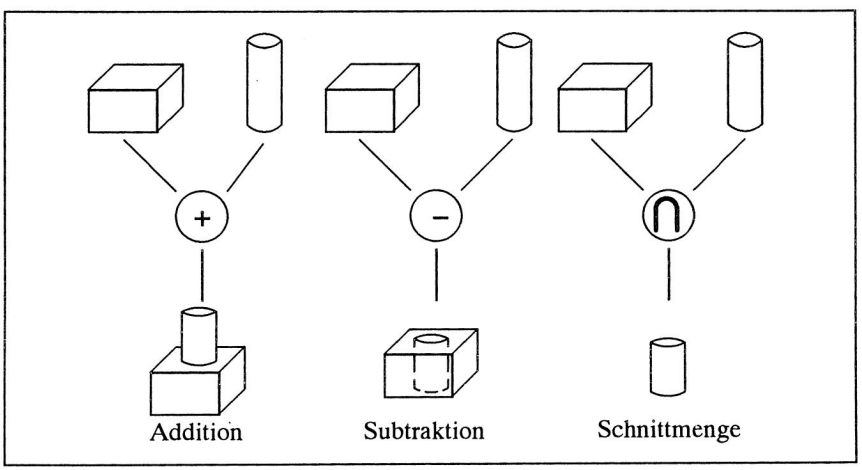


Abb. 30 Darstellung eines technischen Objektes durch eine CSG-Modell

Die Effizienz von Suchverfahren auf Geometrieprimitiven sowie auf Verknüpfungsoperationen wie Schnitt lassen das CSG-Modell für Anwendungen in der Fertigungsplanung zerspanender Bearbeitungsvorgänge geeignet erscheinen. Die Analogie des Schnittes eines Raumprimitivums (Werkzeug) mit einem Ausgangsobjekt (Werkstück) mit dem technischen Zerspanungsvorgang ist hier das maßgebende Kriterium. Jedoch ist hierfür eine geeignete Modellgenerierungsstrategie (Schnittbildung) nötig. Die mehrdeutige (1:n) Beziehung zwischen einem technischem Objekt und möglichen CSG-Modellen erschweren effiziente und stabile Algorithmen.

In der Historie der CAD-Entwicklung haben die CSG-Modelle nicht den Verbreitungsgrad gefunden, der den B-Rep-Modellen zuteil wurde. In der Entwicklung der Feature-Modellierungssysteme kann jedoch eine teilweise Rückbesinnung auf die Vorteile semantisch reichhaltigerer Primitive gesehen werden.

5.1.3 Andere Geometriemodelle

Neben den in den Kapiteln 5.1.1 und 5.1.2 beschriebenen Modellprinzipien existieren noch eine Reihe weiterer Modellschemata, die jedoch wenig verbreitet sind und damit hier nicht von herausragendem Interesse sind.

Zu diesen Zählen die **Octrees** /176/, ein Repräsentationsprinzip das eine Aufteilung eines würfelförmigen Raumbereiches durch rekursive Teilung des Ausgangsraumes durch Teilungsebenen in allen drei Raumrichtungen vornimmt. Die Rekursion endet bei Erreichen einer vorgegebenen Strukturtiefe. Die einzelnen entstehenden Raumbereiche werden durch die Eigenschaft der teilweisen, vollständigen oder nicht vorhandenen Füllung ihres Beschreibungsvolumens gegenüber dem Beschreibungsobjekt charakterisiert. Vorteile der Darstellung sind einfache Algorithmen bei Vereinigung, Schnitt, Durchschnitt und Volumenberechnung von Objekten, sowie deren Translation, hingegen sind Rotation von Objekten und die Analyse von Objekteigenschaften wie Oberflächeneigenschaften oder die Generierung von Fertigungsdaten problematisch oder ungelöst. STEP unterstützt in den Geometriemodellen des Parts 42 keine Octrees.

Das Modellschema der **Zellzerlegung** (spatial decomposition), einer Spezialisierung der Punktmengendarstellung, zerlegt einen darzustellenden Körper in Grundelemente wie Quader, Zylinder und Kegel aber auch spezielle, anwendungsspezifische Grundkörper sind hierbei zugelassen. Die einzelnen Grundkörper berühren sich nur an einzelnen Flächen, Kanten oder Punkten; Schnitte zwischen ihnen sind nicht zugelassen. Schwierigkeiten dieses Darstellungsschemas treten bei parametrisch darzustellenden Flächen auf. Der Zugriff auf einzelne topologisch/geometrische Elemente ist durch die nicht explizite Darstellung schwierig. In den STEP Geometrieschemata werden Zellzerlegungsmodelle nicht unterstützt.

Translations- und **Rotationsmodelle** stellen eine Art nicht evaluierte Modellrepräsentation dar. Die zu repräsentierenden Körper werden durch Rotation oder Translation einer Fläche oder eines offenen oder geschlossenen Kantenzuges erzeugt. Diese sogenannten Sweep-Modelle sind hinsichtlich ihres Anwendungsbereiches auf translations- oder rotationssymmetrische Objekte beschränkt. Reine Sweep-Modelle existieren heute kaum noch. Aus Sweep-Operationen hervorgegangene Objekte werden zumeist evaluiert in B-Rep-Modelle überführt.

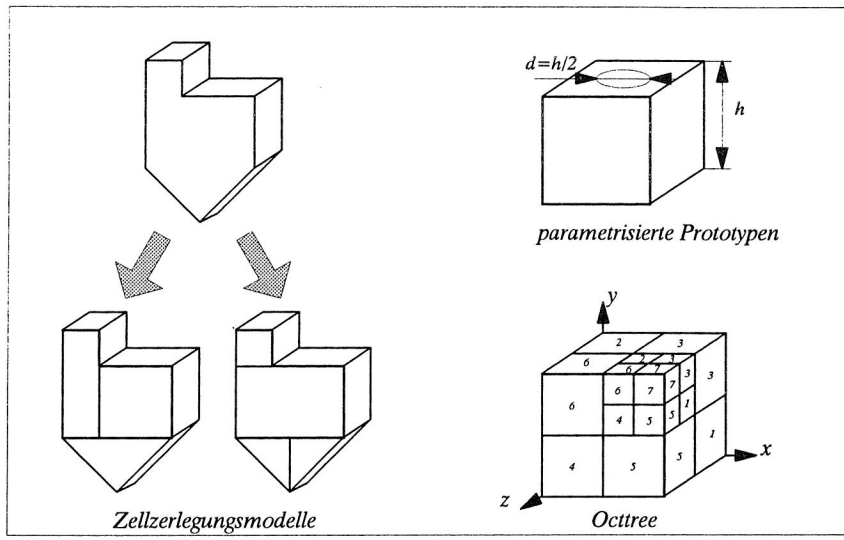


Abb. 31 Alternative Geometriemodellprinzipien

Parametrisierte Prototypen stellen Objekte durch anwendungsbezogene Objekttypen dar, die in den variierbaren Parametern veränderlich sind, und durch Ausprägungen dieser Parameter Instanzen der Prototypen definieren. Dieser Repräsentationsmechanismus orientiert sich an dem Ähnlichkeitsprinzip der Individuen in einem Teilefamilienspektrum. Anwendungen dieser Darstellungsart sind oftmals wenig primär geometrisch orientiert, sondern liegen in der Variantenkonstruktion bzw. in der Bereitstellung von (Norm-)Teilekatalogen. Die geometrische Ausprägung wird hierbei durch Instantiierung aus charakterisierenden Aktualparametern einer Konstruktions- bzw. Instantiierungsfunktion gewonnen. Die eigentliche geometrische Repräsentation wird durch andere Darstellungsprinzipien erreicht, die zu den parametrisierten Prototypen in einer Art Master-Slave-Verhältnis stehen.

Eine der neuesten Entwicklungen in CAD Modelltechnologie zählen verschiedene Arten von **parametrischen** bzw. **parametrisierbaren Modellen**. Innerhalb eines parametrisierbaren Modells lassen sich prinzipiell alle numerischen Charakteristika sowie darüberhinaus geometrische-numerische Beziehungen durch Angabe von Termen beschreiben. Elemente der Terme können dabei Variablen sein, die ein geometrische Eigenschaft beschreiben oder aber wiederum bereits definierte Terme. Durch die vorgegebenen Terme entsteht ein Gleichungssystem, das durch Gleichungslösungssysteme bearbeitet wird. Die Lösbarkeit eines solchen Gleichungssystems stellt dabei die bekannten Anforderungen die jedoch übertragen an geometrisch definierte Relationen schnell unüberschaubar werden.

Parametrisierbare Modelle stellen im Wesentlichen eine neue Art der Kommunikation zwischen einem Anwendungssystem und einem internen Modell dar. Die interne Mo-

dellrepräsentation wird dabei von einem B-Rep- oder einem CSG-Modell übernommen. Das parametrisierbare Modell besteht aus einem Gleichungssystem und einem Geometriemodell und steht zu der evaluierten internen Geometrierepräsentation in einem Master-Slave-Verhältnis. Hinsichtlich der Modellierungsfreiheit lassen sich parametrisierbare Modelle in topologieerhaltende und topologieverändernde Modelle unterscheiden. Während die topologieerhaltenden Modelle letztlich eine extern (für das topologisch-geometrische Modell) definierte Beeinflußbarkeit der geometrischen Metriken darstellt und damit leicht als Mastermodells eines B-Rep-Modells realisiert werden kann, stellen topologieverändernde parametrische Modelle eine Möglichkeit der Beeinflussung der Aktualparameter einer Modellentstehungshistorie dar.

5.2 STEP-Prozessoren zwischen CAD-Systemen

Zur Entstehungszeit sind nach Standardisierung des ersten Teils der ISO-10303 im September 1994 von verschiedenen Herstellern Prozessoren für die Anwendungsprotokolle 201 /60/ und 203 (Configuration Controlled Design) angekündigt bzw. erste Versionen ausgeliefert. Wie weiter in Kapitel 6 verdeutlicht wird, stellen die Anwendungsprotokolle die Umgebung für Implementierungen von STEP-kompatiblen Anwendungen dar. Damit wird deutlich, daß Implementierungen von Prozessoren z.B. für AP 201 und 203 *unterschiedliche* Anwendungen sind und nicht notwendigerweise kompatibel zueinander sein müssen. Dies bedeutet konsequenterweise, daß *alle* STEP-Prozessoren, die sich auf unterschiedliche AP's beziehen prinzipiell untereinander paarweise inkompatibel sind. (Dies bedeutet nicht automatisch, daß eine STEP-Datei, die konform zu einem AP203 geschrieben wurde, nicht von einem AP214 Prozessor gelesen werden kann). Die bisher 39 definierten APs führen also zu 39 unterschiedlichen Prozessoren. Auf einem neuen Niveau werden damit eine Vielzahl inkompatibler "STEP-Prozessoren" definiert. Diese Problematik läßt sich unter dem dem Begriff "Interoperabilität" subsumieren. Diese Problematik wird in Kapitel 6.2 weiter detailliert.

Zum Entstehungszeitpunkt der Arbeit sind von verschiedenen Herstellern Prozessoren für die Anwendungsprotokolle 201, 203 und 214 zumindest in Vorversionen verfügbar. Obwohl zu diesem Zeitpunkt die Spezifikation des AP214 in wesentlichen Teilen noch nicht stabil ist, haben CAD Systemhersteller vornehmlich für den deutschen Markt Prozessoren für dieses AP entwickelt. Die Problematik der Interoperabilität zwischen den APs 203 und 214 ist dabei zumindest erkannt worden. Entwicklungsziel des AP214 ist es dabei, ein Anwendungsprotokoll zu schaffen, daß vollkommen aufwärtskompatibel zum AP203 ist. Dies bedeutet, daß STEP-Austauschdateien, die nach dem AP203 geschrieben wurden, von Prozessoren, die nach dem AP214 implementiert wurden, gelesen werden können. Eine Kompatibilität in umgekehrten Sinne wird nicht unterstützt.

Hersteller	*System*	*AP203 Konformitätsklassen*						*AP214 Konformitätsklassen*	
		CC1	*CC2*	*CC3*	*CC4*	*CC5*	*CC6*	*CC1*	*CC2*
Hewlett-Packard	*SolidDesigner 3.5*						*X*	*X*	
CAP-debis	*CATIA V4.1.4*						*X*	*X*	
Computervision	*CADDS 5*		*X*		*X*		*X*	*X*	
Parametric Technology	*Pro/ENGINEER*		*X*		*X*		*X*	*X*	
EDS	*Unigraphics 10.5*	*X*	*X*			*X*	*X*	*X*	
SDRC	*IDEAS Masters Series*	*X*	*X*		*X*		*X*	*X*	
Autodesk	*AutoCAD*						*X*	*X*	
Dassult Systems	*CATIA V4 FR5*	*X*	*X*		*X*	*X*	*X*	*X*	
Intergraph	*EMS 3.1*	*X*	*X*		*X*		*X*	*X*	
Tecnomatix	*ROBCAD*						*X*	*X*	

Tab. 5 Verfügbare STEP-Prozessoren für CAD/CAE-Systeme nach ProSTEP

Wie der Tabelle zu entnehmen ist, sind gerade CAD-Hersteller die treibende Kraft der Definition und Realisierung von STEP-Prozessoren. Obwohl auch SW-Hersteller z.B. aus dem Bereich der EDM-Systeme oder kommerzieller Software großes Interesse an STEP zeigen, sind Implementierungen aus deren Umfeld noch nicht verfügbar. Eine Begründung kann in der schwächeren Unterstützung von nichtgeometrischer Information gesehen werden, wie sie bereits in Kapitel 2 angesprochen wurde.

Die Realisierung von Prozessoren für CAD-Systeme ist im wesentlichen an die Übertragung geometrischer Information aus den Geometriemodellen der CAD-Systeme bzw. deren interner Geometriedarstellung gebunden. Im folgenden soll die Implementierung von STEP Post- und Preprozessoren für Geometriemodellierungssysteme dargestellt werden. Die beiden benutzen Modelliersysteme Parasolid (Shape Data Ltd.) und ACIS (Spatial Technology) werden in einer Vielzahl von kommerziell verfügbaren CAD-Systemen eingesetzt. Die Realisierungen können damit beispielhaft für industriell relevante STEP-Prozessoren angesehen werden.

5.2.1 STEP-Partialmodelle zur Geometriebeschreibung

In den Partialmodellen des Parts 42 der ISO 10303 werden topologische und geometrische Elemente sowie darauf aufbauend Geometriemodelle definiert, die eine hohe Allgemeingültigkeit besitzen und damit für beliebige Anwendungsprotokolle benutzt werden sollen. Für die folgenden Kapitel 5.2.2, 5.2.3 und 5.3.1 ist das Verständnis dieser Geometrie- und Topologiemodelle von eminenter Bedeutung, so daß im folgenden eine kurze Vorstellung dieser Modelle erfolgen muß.

Der Part 42 besteht aus drei Schema, die zur Beschreibung von geometrischen Elementen (*geometry_schema*), zur Beschreibung der Topologie (*topology_schema*) und zur Beschreibung von geometrischen Grundkörpermodellen (*geometric_model_schema*) dient.

Die Sachverhalte, die das Geometrieschema des Teiles 42 beschreibt, sind einerseits generische Primitive wie 'point' (allgemeiner Punkt), 'direction' (Vektor mit Position), 'vector' (Vektor), 'surface' (Oberfläche), andererseits aber auch sehr implementierungsnah definierte Konstrukte wie 'trimmed_curve' (getrimmte Kurve) oder 'point_replica' (Replikat eines explizit definierten Punktes).
Einerseits wird in der Geometriedefinition versucht mathematisch-geometrisch begründete Ähnlichkeiten typisierend abzubilden wie das durch die Spezialisierungen der Kegelschnitte (*conic*) in der Kreiskurve (*circle*), der Ellipse (*ellipse*), der hyperbolischen (*hyperbola*) und der parabolischen (*parabola*) Kurve geschieht. Daß auch die gerade Kurve durch einen Kegelschnitt beschrieben werden kann, wird in diesen Definitionen ignoriert. Die Definitionen zeigen offenbar bereits hier die Bemühungen um einen Kompromiß zwischen logischer Konsequenz und implementierungstechnischen Orientierung.

Das Topologieschema stellt Elemente zur Verfügung, um ein topologisch definiertes Geometriemodelle definieren zu können. Dabei müssen Elemente mit

- geometriefreier und
- geometriegebundener

Ausprägung unterschieden werden. Zu den Vertretern erster Klasse gehören im Part 42 u.a. *'vertex'* (Scheitelpunkt), *'edge'* (Kante) und *face* (Fläche), zu letzterer Spezialisierungen obiger in *'vertex_point'*, *'edge_curve'* und *'face_surface'*. Die Zuordnung geometrischer Elemente zu den geometriegebundenen Topologieelementen wird jeweils durch ein Attribut einer geometrischen Entität vollzogen. Daß darüberhinaus jedes geometriegebundene Topologieelement jeweils auch über die Subtypdefinition des *geometric_representation_item* als ein geometrisches Element betrachtet wird, dürfte dabei nur als eine unglückliche Modellbildung betrachtet werden. (Es wird für diese nur ein Attribut "name" vererbt.)

Die Modellbildung der Elemente innerhalb des Parts 42 beinhalten hoch strukturierte Konstrukte wie sie in den Geometriemodellen (*geometric_model_schema*) benutzt werden. Aufgrund der in den STEP Partialmodellen eigenen Modellierungscharakteristik werden viele Charakteristika eines Objektes nicht explizit in den Eigenschaften, die durch die Attribute ausgedrückt werden niedergelegt, sondern implizit in sogenannten 'informellen Vorschlägen' (Informal propositions) festgelegt. Der Verzicht auf die mögliche, weitgehend typisierte Spezifikation einer Objektbeschreibung stellt ein wesentliches Hindernis für eine ausdrucksstarke Implementierung dieser Objekte dar. Weiteres hierzu und vorgestellte Lösungen sind in Kapitel 6.1.1 verdeutlicht.

Die Partialmodellschemata des Parts 42 der ISO 10303 stellen den gemeinhin akzeptierten Modellbereich aller STEP Modelle dar. Hinsichtlich ihrer Modellkonzeption sind jedoch einige Modellierungscharakteristika unbefriedigend. Obwohl nicht primär implementierungstechnisch orientiert, sind viele Eigenschaften dieser Modelle spezifischen Implementierungen geometrieverarbeitender Systeme nachempfunden. Diese implementierungstechnischen Kompromisse wurden oftmals zu Ungunsten mathematischer Äquivalenz oder modellierungstechnischer Klarheit getroffen. Andererseits ist auch die Implementierungsorientierung nicht immer einer softwaretechnischen Modularisierung zugänglich. Das Prinzip der Spezialisierung gilt nicht für alle Objekte, die in den Geometrieschemata abgebildet wurden.

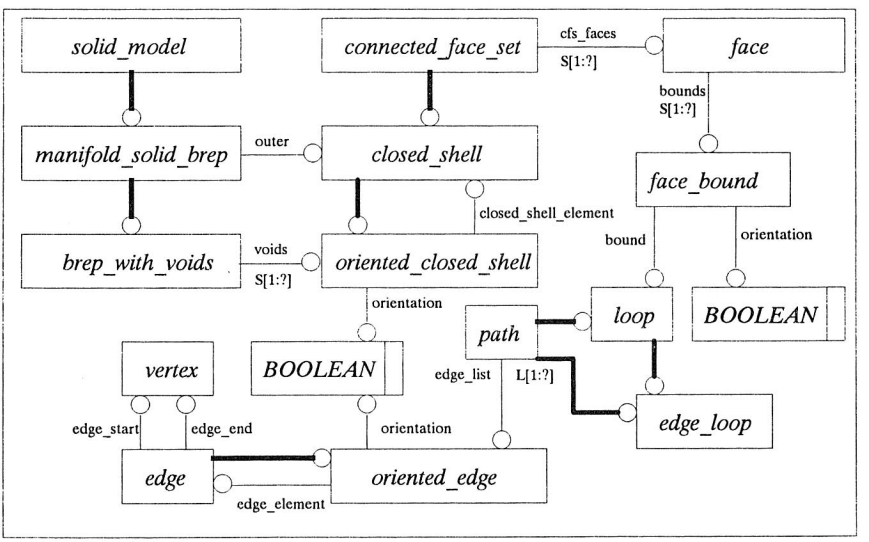


Abb. 32 Topologisches Modellschema des manifold_solid_brep Konstruktes aus dem Part 42 der ISO 10303 in EXPRESS-G

In Abb. 32 wird das teilweise vereinfachte topologische Modell des mannigfaltigen Volumenkörpers (*manifold_solid_brep*) in einem EXPRESS-G Diagramm dargestellt. Dabei fällt auf, daß auf dieser Modellierungsebene die Möglichkeiten zur Strukturierung dieses Modells nur teilweise genutzt wurden. Es ist aus diesem Diagramm ersichtlich, daß z.B. für die Endpunkte (*vertex*) von Kanten (*edge*) keine inverse Beziehung zu den Kanten formuliert wurde[14]. Aus Sicht eines Implementierungsnahen Modells ist dies als Nachteil aufzufassen. Wie später in Kapitel 5.2.3 gezeigt wird bedeutet dies jedoch nicht, daß die Abbildbarkeit des STEP-Geometriemodells auf andere Modelle nicht möglich ist.

14. *Dies gilt offensichtlich für alle Beziehungen zwischen direkt verbundenen topologischen Elementen.*

Hingegen stellt das vorgegebene STEP kein implementierungstechnisch optimales Modell dar.

Obwohl in den Geometriepartialmodellen (Teil 42) von STEP CSG-Modelle prinzipiell darstellbar sind, ist diese Technik jedoch wenig einsetzbar, da sich die Menge der unterstützten Primitiven auf die Trivialfälle Quader, Zylinder, Kegel und Torus beschränkt. Die Nutzbarkeit dieses Modellprinzips ist jedoch von der Darstellbarkeit auch komplexerer Körper z.B. durch B-Rep-Modelle abhängig.

5.2.2 Postprozessor für Parasolid

Der Geometriemodellierer Parasolid wird von der Firma Shape Data Ltd. entwickelt und vertrieben /202/,/203/. Er wird z.B. in den CAD-Systemen Unigraphics (EDS), Sigraph-3D (Siemens) und I-CAD (I-CAD) sowie in FEM-Systemen (Patran) eingesetzt.

Parasolid ist der Nachfolger des weithin bekannten Modellierers Romulus, der den Erfolg der B-Rep-Modellierer in CAD-Systemen weitgehend mitbegründet hat. Im Gegensatz zu Romulus, der in FORTRAN implementiert worden war, ist Parasolid in einem C-Dialekt realisiert. Das interne Modellschema von Parasolid ist ein exakter Boundary-Representation Modellierer, der eine Vielzahl von unterschiedlichen Typen geometrischer Elemente unterstützt. Dazu zählen im Flächenbereich analytische (ebene, zylindrische, konische, sphärische und torodiale) Darstellungen, sowie Approximationsdarstellungen nach beliebigen mathematischer Formulierungen (Splines, NURBS, Bezier) sowie Importflächen. Bei Kurvenmodellen unterscheidet Parasolid ebenfalls in analytische (gerade, kreisförmige, elliptische) und approximative Darstellungen.

Das topologische Modell von Parasolid entspricht weitgehend dem in Kapitel 5.1.1 dargestellten allgemeinen Schema. Das Softwarepaket, das den Modellierer Parasolid darstellt besitzt zwei Schnittstellen, von denen eine zur Manipulation der zu modellierenden Objekte benutzt wird (das sogenannte KI, Kernel Interface) sowie eine Schnittstelle, die zur Anbindung von Systemen zur grafischen Darstellung der modellierten Objekte (Graphical Output, GO) sowie zur Anbindung von Systemen zur Hauptspeicherverwaltung und zur Modelltransformation auf Hintergrundspeicher, insbesondere Dateisysteme auf Festplatten über das sogenannte 'Frustrum' dient.

Das Kernel Interface stellt das API zur Modellmanipulation des Modellierungsobjektes dar und besteht aus etwa 220 Funktionen, die über ein einheitliches Schema die Kommunikation mit dem Modelliersystem ermöglichen. Das Kernel Interface von Parasolid zeigt sich prinzipiell als eine transaktionsorientierte Schnittstelle im Sinne des B-Rep-Schemas. D.h. jede Interaktion eines Anwendungsprogrammes über das API transformiert über einen modellverändernden Funktionsaufruf eine gültige B-Rep-Darstellung des Modellierungsobjektes wieder in eine gültige B-Rep-Darstellung. Die einzelnen API-Funktionsaufrufe von Parasolid stellen damit eine Art Transaktionsschale über die

Modellierungsobjekte dar, wie das ähnlich für Datenbanksysteme nach dem ACID-Prinzip gilt.[15]

Parasolid ist in der Lage, Objekte in verschiedenen internen Darstellungsarten zu halten, die durch das Modellschema dargestellt werden können, er versucht dabei immer das topologisch höchstwertigste Modellschema zu erreichen. Z.B. kann eine verbundenen Menge von Kanten als ein Linienmodell dargestellt werden. Wird der Linienzug geschlossen, erzeugt Parasolid automatisch ein Loop. Parasolid ist daher in der Lage prinzipiell in der Lage, Objekte durch unterschiedliche Modellschemata darzustellen und zu behandeln.

Für den realisierten Postprozessor wurde jedoch nur eine Teilmenge aller darstellbaren Modellschemata behandelt. Im Hinblick auf das realisierte 3D-ECAD-System (Kapitel 7.3) sind hierfür nur Modelle interessant, die geometrisch und topologisch Mindestanforderungen erfüllen, die nur mit Sicherheit durch Solid-Modelle des Parasolid erfüllt werden. Die präferierte Arbeitstechnik in CAD-Systemen sorgen jedoch zumeist für die Darstellung von Objekten in Solid-Repräsentation, so daß dies keine wesentliche Einschränkung der Realisierung darstellt. Die Erweiterung der Implementierung auf bisher nicht unterstützte Modelltypen ist darüberhinaus bereits durch die Behandlung untergeordneter Geometrie- und Topologieelemente nur eine implementierungstechnische Erweiterung ohne weitere wesentliche Problemlösungskomponente.

Das Konzept der Implementierung des Postprozessors basiert auf der Top-Down, Depth-First, Upward-Collection Prinzip, einer rekursiven Traversierung der topologischen Parasolid-Elemente (Abb. 33). Da innerhalb eines korrekten Parasolid Objektes die Kardinalität der topologisch untergeordneten Elemente mindestens eins ist (Ausnahmen siehe unten), kann ein einfacher Traversierungsalgorithmus aufgebaut werden, der auf diesem Fakt basiert. Das Schema dieses Traversierungsschemas lautet für jeden topologischen Elementtyp (vereinfacht dargestellt in C-Syntax):

```
/* Identifikation der jeweils topologisch verbundenen Elemente TYTO?? */
IDCOEN (&element, &TYTO??, &elementliste, &elementanzahl, &ifail);
/* Extraktion der Elemente aus der Liste */
GTTGLI (&elementliste, &eins, &elementanzahl, subelement, &ifail);
for (i = 0; i<= elementanzahl-1; i++) {
        ENENTY(subelement[i], &vier, types, ntypes, &ifail);
        ...
}
```

Dabei wird mit der Konstanten TYTO?? der jeweils gesuchte Typ des topologischen Elementes selektiert. Beispielsweise wird für ein Element vom Typ 'body' (TYTOBY) der verbundene Typ 'shell' (TYTOSH) qualifiziert.

---

15. Der Vergleich des Parasolid API mit einem transaktionsorientierten Datenbanksystem bezieht sich wie dargestellt nur auf die Fähigkeit zur Erhaltung der geometrischen und topologischen Integrität des internen Modells. Die anderen Eigenschaften des ACID-Prinzips werden hierdurch natürlich nicht unterstützt.

Die Generierung der STEP-Austauschdatei wird während der Traversierung des Parasolid-Modells nicht sofort vorgenommen, hingegen werden die Instanzen der Modellentitäten prozeßintern erzeugt und nach abgeschlossener Traversierung mit Hilfe des in Kapitel 3.3.3 erläuterten Austauschdateiformatierers auf eine beliebige Datei geschrieben. Dies ermöglicht einerseits die nachträgliche Veränderung bereits erzeugter Instanzen sowie durch Modularisierung eine Trennung der funktionalen Aufgabenbereiche des Programmsystems.
Beim Aufstieg aus einer tieferen Elementhierarchie sind die abzubildenden Elemente erzeugt und deren Bezeichner bekannt. Topologisch höherstehende Elemente die Verweise auf die sie bestimmenden untergeordneten Elemente benötigen, können beim Erzeugen auf diese bereits existierenden Elemente (bzw. deren Bezeichner) verweisen, ohne Vorwärtsverweise vornehmen zu müssen.

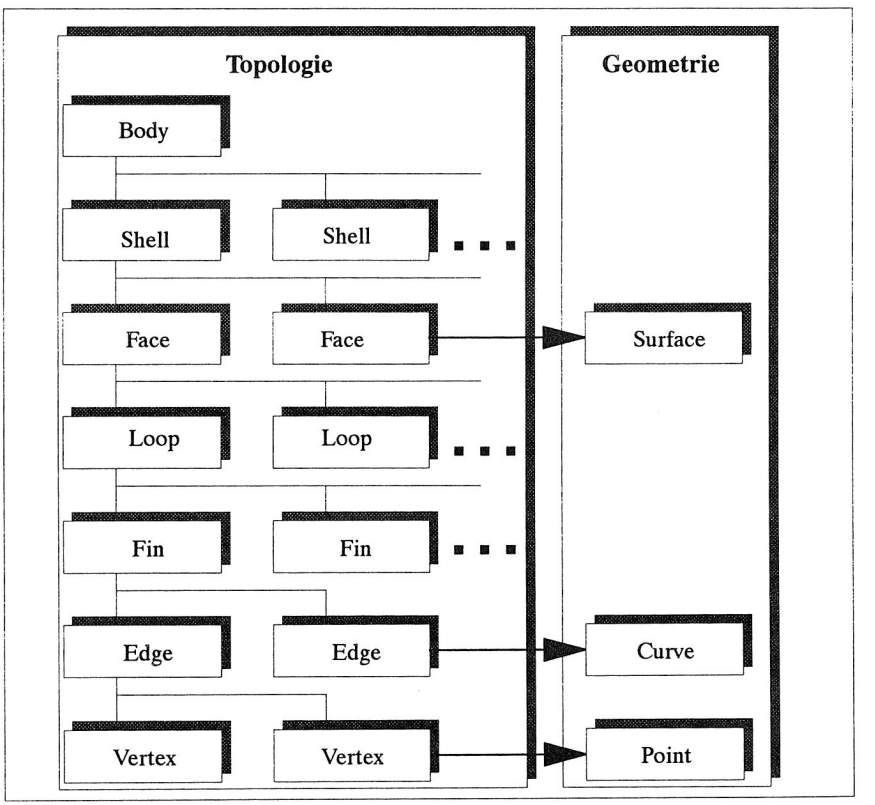


Abb. 33 Top-Down Traversierung der topologischen Parasolid-Elemente

Für jedes traversierte topologische Element, dem ein geometrisches Element zugeordnet ist (dies gilt für 'face' mit 'surface', 'edge' mit 'curve' und 'vertex' mit 'point'), wird das

geometrische Element erfragt und dessen Eigenschaften bestimmt. Der Traversierungsalgorithmus traversiert immer zunächst in die Tiefe zum topologisch nächsttieferen Element, bevor die geometrische und danach die topologische Information eines Elementes bestimmt wird. Dies ist aus dem Abbildungsmechanismus der EXPRESS-Entitäten auf die Austauschdatei nach Part 21 der ISO 10303 nötig. Beispielhaft sei die Abbildung eines topologischen Punktes (*vertex*) samt seiner geometrischen Repräsentation in einem geometrischen Koordinatenpunkt gezeigt:

In Teil 42 der ISO 10303 werden die Elemente VERTEX, VERTEX_POINT, POINT und CARTESIAN_POINT definiert:

```
ENTITY vertex
SUBTYPE OF (topological_representation_item);
END_ENTITY;

ENTITY vertex_point
SUBTYPE OF (vertex, geometric_representation_item);
        vertex_geometry   :        point;
END_ENTITY;

ENTITY point
SUPERTYPE  OF  (ONEOF(cartesian_point,  point_on_curve,  point_on_surface,
point_replica, degenerate_pcurve))
SUBTYPE OF (geometric_representation_item);
END_ENTITY;

ENTITY cartesian_point
SUBTYPE OF (point);
        coordinates       :        LIST [1:3] OF length_measure;
END_ENTITY;
```

Ein Punkt, der z.B. einen Endpunkt einer Kante darstellt muß dann etwa wie folgt in einer Übertragungsdatei beschrieben werden:

```
...
#7=CARTESIAN_POINT('Punkt_1',(0.0,0.0,0.0));
#8=VERTEX_POINT('Vertex_1',#7);
...
```

Da ein topologisches Element immer einen Verweis auf ein geometrisches Element besitzt, ist es sinnvoll, zunächst das geometrische Element zu erzeugen und mit dem dann bekannten Instanznamen des geometrischen Elementes, das topologische Element zu erzeugen. Damit ist es im Allgemeinen nicht nötig, Vorwärtsverweise auf noch nicht bekannte Elemente zu erzeugen (Eine Ausnahme ist die Beschreibung von Kurven, die aus dem Schnitt zweier Flächen hervorgehen, s.u.).

Die Zuweisung von Namen von Instanzen innerhalb einer Übertragungsdatei ist prinzipiell frei (nach ISO 10303 muß es sich um eindeutige Zeichenketten handeln, die als Festkommazahlen interpretierbar sein müssen), hinsichtlich der Implementierung des Traversierungsalgorithmus wurde jedoch eine spezielle Vorgehensweise gewählt, um Implementierungsrandbedingungen sowie Performanzaspekte zu berücksichtigen:

In einer von Parasolid verwaltete Liste werden die Elementbezeichner aller abgearbeiteten Parasolid-Elemente angehängt. Die Position des Elementes in der Liste ist damit mit dem Instanznamen in der Übertragungsdatei identisch. Für Instanzen, die in die Übertragungsdatei geschrieben werden sollen, für die jedoch kein Parasolid-Äquivalent existiert wird ein Dummy-Element an die Liste angehängt. Damit kann jederzeit durch Untersuchen der Liste bestimmt werden, wieviele Instanzen bereits erzeugt wurden, ob ein Parasolid-Element bereits abgearbeitet wurde und welcher Instanznamen, entsprechend dem Listenplatz, diesem Element zugewiesen wurde.

Eine Problem der Top-Down, Depth-First Traversierung des Parasolid-Modellschemas tritt bei der Behandlung sogenannter Schnittkurven auf. Die geometrische Beschreibung der Kurven, die aus dem Schnitt zweier Flächen hervorgehen und nicht analytisch durch die in der Parasolid Modellwelt unterstützten Kurventypen beschrieben werden können, erfolgt durch Verweis auf die beiden zum Schnitt herangezogenen Flächen.

Da für die im AP203 (und damit vermutlich in allen Geometriebeschreibenden APs) die Beschreibung über die in Part42 definierte INTERSECTION_CURVE nicht erlaubt ist, wird diese Kurve mittels einer B_SPLINE_CURVE (bzw. einer Spezialisierung hiervon) beschrieben. Da Parasolid jede seiner analytisch oder anderweitig repräsentierten Kurven durch Spline-Darstellungen wiedergeben kann, ist in eine derartige Kurve durch eine Spline-Kurve wiederzugeben und in die STEP-Darstellung zu übertragen.

Die primären Objektdarstellungen in Parasolid sind Volumenkörper, die auch Hohlbereiche (*voids*) aufweisen können. Die für die Abbildung gewählte Zielstruktur ist das im Part 42 definierte Geometriemodell *manifold_solid_brep.* Der Traversierungsalgorithmus detektiert innerhalb eines Parasolid Volumenobjektes die Anzahl der internen Hohlbereiche. Ist deren Anzahl größer gleich eins, so wird ein spezialisierter Untertyp des *manifold_solid_brep* instantiiert, nämlich das *brep_with_voids*.

Im Wesentlichen kann das topologische und geometrische Modellschema von Parasolid direkt auf das geometrische Partialmodell der ISO 10303 abgebildet werden. STEP verfügt hingegen sogar über eine größere Typenvielfalt von topologischen und geometrischen Elementen. Dies ist für einen Postprozessor irrelevant, solange jedes Element des internen Modellschemas auf ein beliebiges Element des STEP-Modellschemas abgebildet werden kann. Auch eine eineindeutige Abbildbarkeit ist nicht nötig: durch den Modellkontext kann zu jedem Zeitpunkt die Semantik eines auch mehrdeutig verwendeten STEP-Modellelementes bestimmt werden. Als Beispiel sei die Abbildung von Kreismittelpunkten als *cartesian_point* in STEP genannt. Kreismittelpunkte in Parasolid werden als Attribute (Tripel von Gleitkommawerten) beschrieben. Ein ausgezeichneter Punkt existiert hierfür im Parasolid Modellschema nicht.

Die Unterschiede zwischen den Modellschemata von STEP und Parasolid sind eher minimal und spielen nur eine untergeordnete Rolle bei der Abbildung der Elemente. So stellt Parasolid einen geschlossenen Kreis sowie eine geschlossene Ellipse als ein Loop dar, welches keine topologischen (und damit auch keine geometrischen) Punkte be-

sitzt. In diesem Fall müssen für die Abbildung in STEP aus der parametrischen Kreis- oder Ellipsendarstellung zwei geometrische Punkte für den Parameterwert u=0 für C(u) und darauf basierend zwei topologische Punkte (*Vertices*) bestimmt werden.

Das STEP Element *advanced_brep_shape_representation,* in dessen Elementliste *manifold_solid_brep* Instanzen eingeordnet werden, besitzt eine Reihe von Zwangsbedingungen, die für ein korrektes Modell gelten müssen. Ohne explizite Überprüfung der bei der Abbildung erzeugten Instanzen kann mit Sicherheit geschlossen werden, daß jeder erzeugbare Datensatz die geforderten Zwangsbedingungen erfüllt, da das interne Modellschema von Parasolid sämtliche Regeln erfüllt, und in vielen Fällen sogar weitaus strengere interne Regeln besitzt.

5.2.3 Preprozessor für ACIS

ACIS ist wie Parasolid ein Geometriemodellierer zur exakten Beschreibung technischer Objekte. Das Softwaresystem wird von den Firmen Three-Space Ltd. und Applied Geometry Corporation entwickelt und der Firma Spatial Technology Inc. /210/ vertrieben. Das Modelliersystem wird in einer Vielzahl von CAD- und CAE-System eingesetzt. Einige der bekanntesten Systeme hierunter sind: SolidDesigner von Hewlett Packard, ICEM von Control Data, Konsys 2000 von Strässle, AutoCAD von AutoDESK (ab Version 13), Microstation Modeler von Bentley/Intergraph und ARIES von ARIES Technology um nur einige zu nennen.
Das Entwicklungsteam von ACIS ist teilweise aus der Entwicklungsteam des Romulus, dem Vorgänger von Parasolid hervorgegangen. Die Ähnlichkeit mancher Modellierungskonzepte zwischen ROMULUS/Parasolid einerseits und ACIS andererseits ist hierauf zurückzuführen. Im weitesten Sinne ist ACIS damit ebenfalls, wenn auch indirekt, ein Nachkomme von ROMULUS.

Die interne Modellstruktur von ACIS ist im Gegensatz zu anderen Modelliersystemen wesentlich weniger eingeschränkt. ACIS macht wesentlich weniger strenge Annahmen über die topologische und geometrische Struktur eines erzeugten Modells. Dies wird von vielen Seiten als ein Fortschritt gegenüber früheren Entwicklungen angesehen. Hintergrund dieser abgeschwächten Konsistenzanforderung ist die Möglichkeit, zwischen Konstruktionsschritten ein nicht konsistentes Modell halten zu können. ACIS stellt über seine API-Funktionen also im Gegensatz zu Parasolid keine Art von Transaktionsmechanismus zwischen konsistenten Modellen bereit. Die Konsistenzsicherung muß explizit durch den Anwendungsprogrammierer erfolgen; ein Zwang hierzu besteht nicht. Mithin können inkonsistente Modelle sogar Sitzungsperioden überdauern.

ACIS ist in C++ (mit Ausnahme der Parameterflächen, diese sind in C) implementiert. Die Modellstruktur von ACIS ist direkt über die Klassendefinitionen und über die in C++ möglichen Zugriffe auf die Attribute der Klassen erreichbar. Hiermit ergibt sich ein sehr

direkter Zugriff auf das Modell selbst. ACIS stellt insgesamt 4 gestufte Ebenen des Zugriffs auf das interne Modell zur Verfügung (Abb. 34).

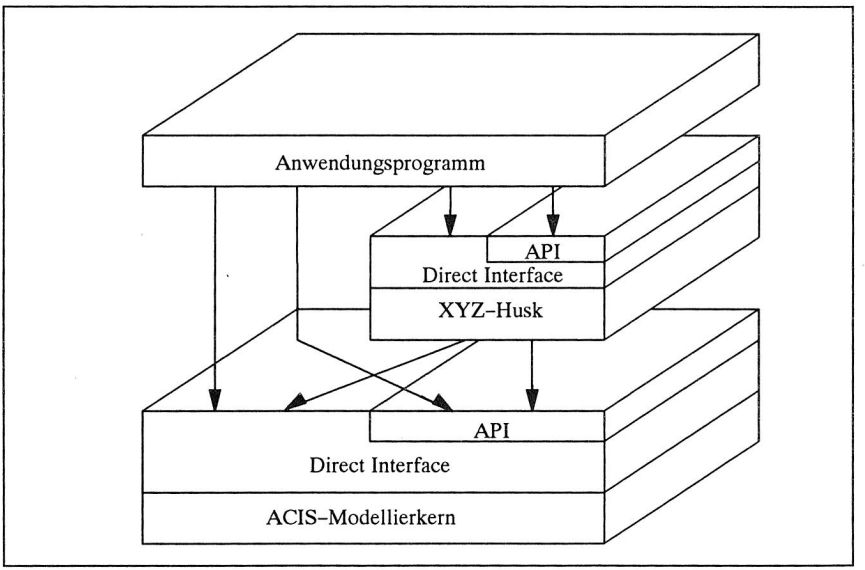


Abb. 34 Zugriffsmechanismen auf Objektbeschreibungen durch ACIS

Neben den direkten Memberfunktionen der ACIS Kernel-Klassen, die zur Erzeugung (Konstruktoren), Vernichtung (Destruktoren) und Strukturbildung unter Verwendung von Basiselementen dienen (*Direct Interface*), bietet ACIS API-Funktionen für semantisch höherwertige Aufgaben, wie geometrische Vereinigung, Schnitt oder Durchschnitt zweier Objekte an. API-Funktionen erlauben es, den internen Modellstatus zwischen Funktionsaufrufen über sogenannte Bulletin-Board Mechanismen zu sichern und zwischen verschiedenen Modellierungszuständen direkt zu wechseln.

ACIS ist konzeptionell durch seinen internen Aufbau weitgehend offen für Erweiterungen auf verschiedenen Niveaus. Zum einen sind die verschiedenen ACIS Klassen über die in C++ realisierten Mechanismen zu erweitern. Darüberhinaus können in sogenannten 'Husks' den Basismodellierer erweiternde Funktionsmodule Anwendungsprogrammierern zur Verfügung gestellt werden, die einerseits wiederum über Klassen (Memberfunktionen) oder auch API-Funktionen angesprochen werden können (Abb. 34).

Die unterstützten geometrischen und topologischen Elemente in ACIS unterscheiden sich etwas von denen, wie sie z.B. in Parasolid möglich sind oder in den STEP definierten Elementen. So kennt ACIS zusätzlich zu den in Abb. 28 gezeigten topologischen Elementen auch noch ein sogenanntes 'Subshell', das eine räumliche, hierarchische

Aufteilung eines räumlichen Objektes vornimmt, im wesentlichen aus Performancegründen, um räumliche Bereiche nur bei berechtigtem Interesse durchsuchen zu müssen.

Darüberhinaus unterscheidet ACIS geometrisch nicht typisierend zwischen Kreis und Ellipse sowie Zylinder- und Kegelfläche. Ausprägungen jeweils beider Typen werden nur über geometrische Eigenschaften determiniert.

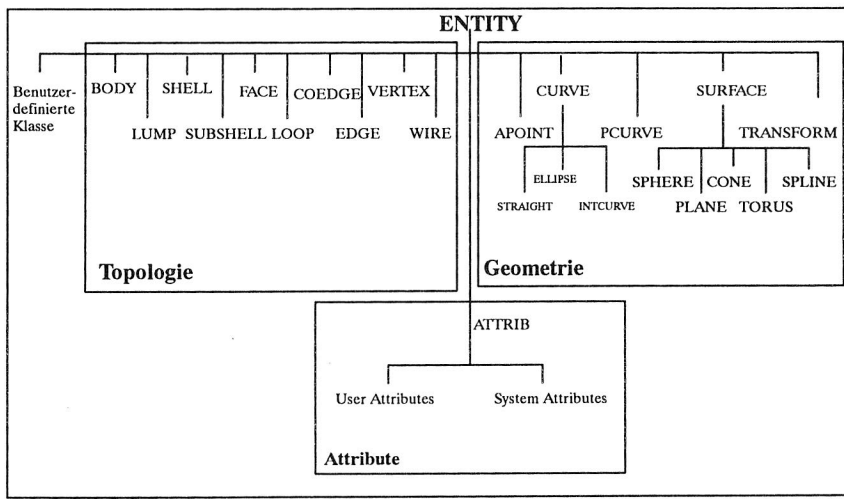


Abb. 35 C++-Klassenhierarchie der Acis internen Klassen (nach /210/)

Die Realisierung eines STEP-File Preprozessors für ACIS wurde unter Berücksichtigung eines AICs vorgenommen. Im Gegensatz zum vorgestellten Postprozessor zu Parasolid (Kapitel 5.2.2), bei dem die modellinhärenten Elemente die Gesamtmenge der zu unterstützenden Austauschelemente darstellt, ist bei einem STEP-Preprozessor prinzipiell jedes innerhalb eines Anwendungsprotokolls mögliche Elementaufkommen zu bewältigen. Die in dem vorgestellten Rahmen von Interesse liegenden Teilmodelle sind in den Anwendungsprotokollen 203, 214 (und voraussichtlich 210) durch die in den AICs der Parts 501 bis 518 definierten Elementumfängen zu finden. Beispielhaft wurde daher das AIC *510* (*advanced_brep_aic*) ausgewählt und durch den Prozessor implementiert.

Realisierungtechnisch stellt der Preprozessor ein spiegelbildliches Analogon zu dem in Kapitel 5.2.2 vorgestellten Parasolid Postprozessor dar. Mit Hilfe eines Top-Down, Depth-First, Upward-Collection Algorithmus wird jede innerhalb einer Austauschdatei erkannte *manifold_solid_brep* Instanz traversiert.

Für jedes der Elemente existiert eine Erzeugungsfunktion, die eine C++Instanz der STEP-Äquivalenz zurückliefert. Die in der Struktur enthaltenen topologischen und geometrischen Elemente werden gemäß Tab. 6 in ACIS erzeugt. Nach der Konstruktion je-

des Elementes wird der (C++) Zeiger auf dieses Element in einem Array unter dem STEP Instanznamen (nach Part 21 Integerwerte) eingetragen. Beim Einstieg in jede Konstruktionsfunktion wird durch Überprüfung des Arrays an der Stelle des Wertes des übergebenen Instanznamens zunächst überprüft, ob die ACIS-Instanz bereits erzeugt wurde, und in diesem Fall die gefundene Instanz zurückgeliefert.

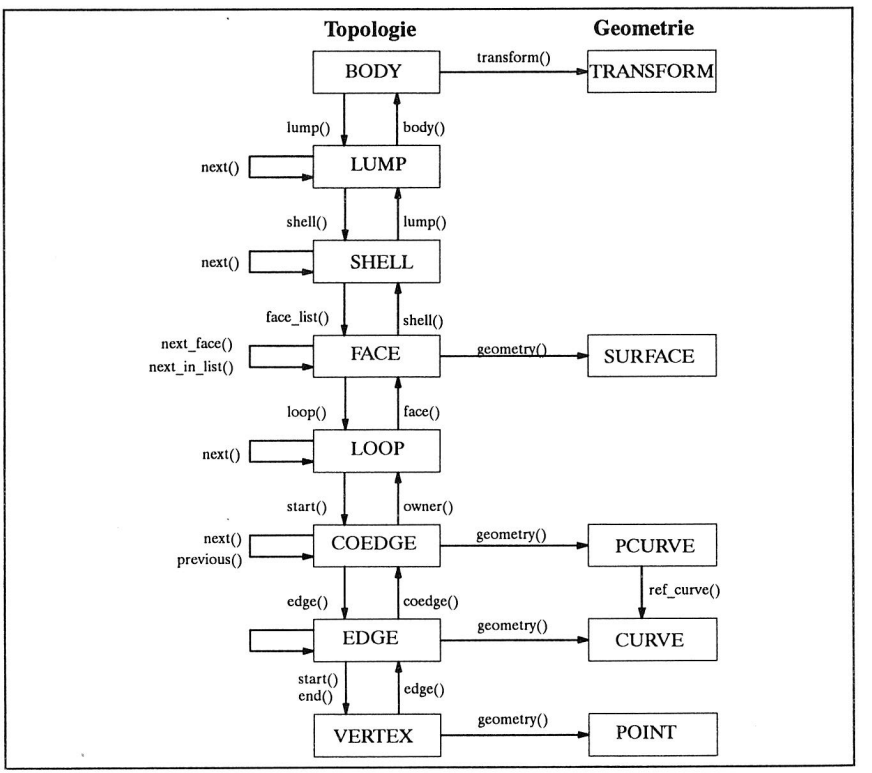


Abb. 36 Interner Modellaufbau von ACIS (nach /210/)

Obwohl die Abbildung zwischen den Elementen der STEP-Geometriemodelle und ACIS Elementen zumeist offensichtlich ist, kann von einer bijektiven Abbildung nicht ausgegangen werden. So existieren STEP-Elemente, die in bestimmten ACIS-Kontexten keine Entsprechung besitzen (z.B. Koordinatenpunkte für lokale Koordinatensysteme *'axis2_placement_3d'*) andererseits aber ausprägungsbestimmte Sonderfälle unterschieden werden müssen, z.B. bei "kreisförmigen" Ellipsen.
Die in Tab. 6 angegebene Gegenüberstellung zeigt nur die für *advanced_brep* möglichen Strukturelemente.

STEP-Geometrie-/Topologieelement (Entität)	*ACIS Geometrie-/Topologieelement (Klasse)*
cartesian_point	*APOINT*
direction	*vector*
axis2_placement_3d	*position–direction–direction*
circle, ellipse	*ELLIPSE*
vertex_point	*VERTEX*
edge_curve	*EDGE*
oriented_edge	*COEDGE*
edge_loop	*LOOP*
face_bound, face_outer_bound	-
plane	*PLANE*
advanced_face	*face*
cylindrical_surface, conical_surface	*CONE*
vector	*vector*
line	*STRAIGHT*

Tab. 6 Abbildungstabelle für Geometrieelemente zwischen STEP-Geometriemodellen und ACIS

Ähnlich wie bei der Erzeugung der STEP-Instanzen der Schnittkurven besteht auch bei der Erzeugung der ACIS Schnittkurven die Problematik auf Flächen verweisen zu müssen, die zum Zeitpunkt der Erstellung der Verweise noch nicht existieren. Da ACIS jedoch keine Anforderung an ein in sich konsistentes Modell stellt, kann innerhalb des Traversierungsalgorithmus zunächst über die Eintragung der Basisflächen hinweggegangen werden. Nach Konstruktion der Schale (*shell*) kann dann ein zweiter Traversierungsalgorithmus die Schnittflächen selektieren und die dann existierenden Flächen als Basisflächen der Schnittkurven eintragen.

Eines der wesentlichen Probleme bei der Erzeugung von Boundary-Representation Modellen aus Datensätzen fremder Systeme liegt in der modellinternen Redundanz und damit möglicher Mehrdeutigkeiten begründet. Da jedes geometrische Element mit Hilfe seiner numerischer Eigenschaften charakterisiert wird, darüberhinaus jedoch zwischen den geometrischen Elementen über die topologische Struktur eine Art geometrische Zwangsbedingung spezifiziert wird, kann unter bestimmten Umständen ein prinzipiell unter Beachtung der STEP Regeln korrektes Modell als inkorrekt aufgefaßt werden.

Beispielhaft stelle man sich eine ebene Fläche vor, die durch einen Kantenzug begrenzt wird. Jeder Punkt der Kante muß ebenfalls ein Punkt der Basisfläche der Kante sein. Der Basispunkt der Linie muß daher zunächst direktes Element der Fläche sein, aber auch die Endpunkte der Kante müssen innerhalb der Fläche zu liegen kommen. Die geometrischen Eigenschaften der Punkte bzw. der Kurven werden jedoch prinzipiell nicht notwendigerweise als ein Parameterwert der geometrischen Fläche repräsentiert, sondern redundant als ein privates Attribut des Punktes bzw. der Fläche. Auch wenn ein Model-

liersystem dem in seinem Kontext korrekten und der Verarbeitungsgenauigkeit genügenden Attributwert eines Punktes bzw. einer Fläche in eine Austauschdatei bringt, so kann es für die Verarbeitungsgenauigkeit eines lesenden Systems zu einer Diskrepanz zwischen einem Basiselement und einem auf ihm definierten Element kommen. Das lesende System wird das Teilelement nicht als geometrisch in dem Basiselement liegendes Element akzeptieren.

Hierbei spielt einerseits die Verarbeitungsgenauigkeit der beteiligten Modelliersysteme, andererseits die Anzahl der in die Austauschdatei geschriebenen Ziffern von Gleitkommazahlen eine Rolle. Die für Geometriemodelle vorgesehene Angabe der Modellauflösung in den Entitäten GLOBAL_UNIT_ASSIGNED_CONTEXT bzw. UNCERTAINTY_MEASURE_WITH_UNIT für Längen und Winkelwerte ermöglicht zumindest die Übertragung der Modellauflösung des schreibenden Systems. Liegt die Modellauflösung des lesenden Systems unter der des schreibenden muß sich das lesende System an die gegebende Auflösung des übertragenden Datensatzes anpassen..
Der Modellierer ACIS läßt hierfür ab Version 2.0 die Veränderung der Modellauflösung zu. Dies wurde in der Prozessorimplementierung genutzt, um die Modelliererauflösung dynamisch anzupassen. Problematisch dabei ist, daß die derart in ACIS-Modelle umgewandelten Datensatze in einer gröberen Auflösung vorliegen. Derartige Modelle können nur in einer Modellierungssitzung erfolgreich weiterbenutzt werden, die ebenfalls die geringere Modellauflösung dieses Modells benutzt. Abhilfe könnte hierbei die Heilung eines derartigen Modells schaffen, bei der die Position von Punkten und der Verlauf von Kurven an die topologischen und geometrischen Ausprägungen der Flächen angepaßt werden.

5.2.4 Datenaustauschszenarien mittels STEP-Prozessoren

Das Prinzip des Datenaustausches zwischen Anwendungssystemen basiert auf dem Prinzip, den hinreichenden Teil eines Datensatzes (Instanzenmenge) zu übertragen, der es ermöglicht mit Hilfe des EXPRESS-Datenmodelles ein vollständiges Abbild des Ursprungsdatensatzes zu erzeugen.

Die Datenelemente die eine redundante Information innerhalb des Datensatzes darstellen, werden nach der Implementierungsmethode für Austauschdateien nicht in die Austauschdatei geschrieben. dabei handelt es sich um berechnete Größen (DERIVE) sowie um über Assoziationen verbundene Datenelemente (INVERSE).

Für die Berechnung einer abgeleiteten Größe ist die Kenntnis der Berechnungsvorschrift für diese Größe nötig, wobei unerheblich ist, ob diese Berechnungsvorschrift als Programmstatements des Preprozessors realisiert wurde oder zur Prozessorlaufzeit aus der EXPRESS-Modellbeschreibung des zugrundeliegenden APs extrahiert wird (Abb. 37). Für inverse Attribute muß über die Assoziation der inversen Beziehung jede

referenzierte Instanz des referenzierten Entitytyps daraufhin untersucht werden, ob die aktuelle Instanz der assoziierten Beziehung genügt.

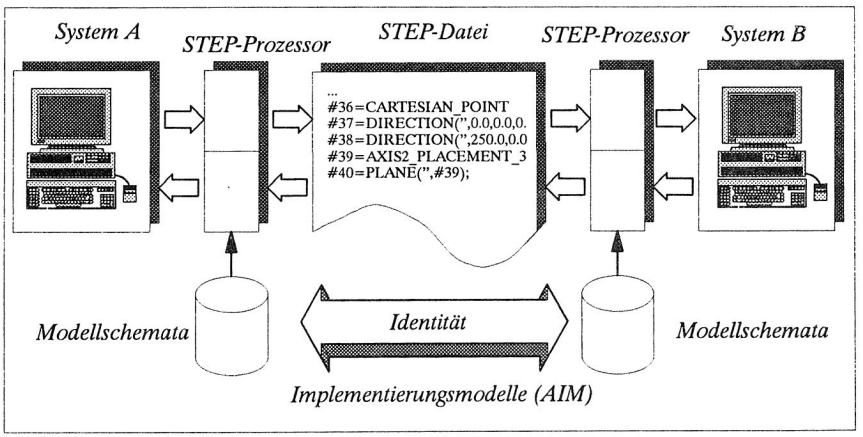


Abb. 37 Austauschprinzipien zwischen Anwendungssystemen auf Basis von Austauschprozessoren

Für eine vollständige Korrektheit eines STEP-Datensatzes im Sinne der Modellbeschreibung aus dem AIM kann darüberhinaus nur die Überprüfung der im AIM angegebenen lokalen und globalen Regeln sorgen. Für alle Entityinstanzen sind diese Regeln zu überprüfen. Sollte bereits eine Regel für eine Instanz verletzt sein, so entspricht der Datensatz nicht mehr einem im Sinne des AIM gültigen Datensatz. Der dann weitere Umgang eines Preprozessors mit dem Datensatz ist nicht vorgegben, jedoch dürfte ein prinzipielles Abweisen des Datensatzes wenig realitätsbezogen sein; andererseits kann eine auch teilweise korrekte Weiterverarbeitung des Datesatzes nicht notwendigerweise sichergestellt werden.

5.3 Datenbankanbindung von ACIS über ein SDAI

Der in den Kapiteln 5.2.4 beschriebene Datenaustausch mittels STEP-Austauschdateien stellt den ersten Fall der Anwendung von STEP dar. Für die Kopplung von Anwendungssystemen, die einen ähnlichen Anwendungsbereich besitzen und über Prozessoren identischer oder kompatibler Anwendungsprotokolle (siehe Kapitel 6.2.1) verfügen, stellt dies einen akzeptablen Weg der Integration in eine unternehmensweite EDV-Landschaft dar. Über Datenverwaltungssysteme (EDM-Systeme) lassen sich die in den Anwendungssystemen erzeugten proprietären Datensätze (CAD-Files) verwalten. Hierfür sind Minimalfunktionalität wie die Vergabe und Überprüfung von Zugriffsberechtigungen, Änderungs- und Freigabezuständen erreichbar. Hingegen ist das Maß der In-

tegration durch die Orientierung auf eben diese proprietären Datensätze und die Ablage der Datensätze außerhalb der Datenbank beschränkt. Letzteres betrachtet die zu verwaltenden Datensätze innerhalb des proprietären Anwendungsmodells als eine atomare Einheit und läßt nur Änderungen auf dieser atomaren Einheit zu. Mit den durch STEP dokumentierten Modellbildung der Produktmodellbeschreibung ist diese monolithische Betrachtungsweise einer hohen Granularität der Produktmodellbeschreibung nicht mehr alleine Sache eines Anwendungssystems, sondern es ermöglicht vielen Anwendungssystemen über die publizierte Modellwelt einen Zugriff und damit die Möglichkeit der Erzeugung und Veränderung auch von Teilen dieses Produktmodells. Wesentlich hierfür ist die Spezifikation einer Zugriffsschnittstelle zu den Produktmodelldaten durch das SDAI.

Die Integration von unternehmensweiten EDV-Landschaften erfordert ein hohes Maß an Integrität, Zugriffsfähigkeit, Sicherheit und Offenheit. Während in dem Bereich der kommerziellen EDV eines Unternehmens Datenbankmanagementsysteme diese hohe Integration sicherstellen, ist in der technisch orientierten Datenverarbeitung eine derartige Konzentration auf ein Datenverwaltungssystem nicht zu erkennen. (Die in jüngster Zeit zum Einsatz gekommenen EDM-Systeme belegen nicht das Gegenteil, stellen sie doch nur eine teilweise Integrationsfähigkeit dar.) Grund hierfür sind einerseits die im technischen Bereich im Vordergrund stehenden Problemlösungsmechanismen technischer Software, die die Datenhaltungsprobleme in den Hintergrund treten lassen. Die zumeist proprietären Dateiformate, die wenig offenen Datenzugriffsschnittstellen und die proprietären internen Modelle der Anwendungssysteme taten ein übriges, um Problemlösungen in diesem Bereich zu erschweren.

Durch das Konzept einer allgemeinen Schnittstelle zu Produktdaten, die in einem bekannten Modellschema vorliegen, können die oben identifizierten Probleme der Integration von technischen Anwendungssystemen gelöst werden. Als ein Kernproblem muß dabei der Zugriff von Anwendungssystemen gelöst werden, die in konventionellen EDV-Landschaften keine Integration in Unternehmensweite Datenbankmanagementsysteme besitzen. Exemplarisch seien hier CAD-Systeme und hiebei wiederum geometrische Modelliersysteme genannt. Gerade geometrische Modelliersysteme stellen ihre Modelle in EDM-Systemen oft nur als grobgranulare Strukturen zur Verfügung, so daß im Gegensatz hierzu bei einer Kopplung eines Geometriemodellierers auf eine Datenbankschnittstelle, wie dies das SDAI darstellt einige neue Problemstellungen identifiziert werden können, die im folgenden erläutert werden.

5.3.1 Konzept zur Kopplung von ACIS auf ein C++ early-binding SDAI

Kopplungen zwischen CAD-Systemen und Datenbanken sind zahlreich in der wissenschaftlichen Literatur erwähnt /95//68/. Kennzeichen all dieser Konzepte oder Realisierungen ist entweder die ausschließliche Verwaltung von CAD-Metainformationen über

das Geometriemodell (Dateinamen, Versionsnamen usw.) in der Datenbank oder die Kopplung zu einer Datenbank deren internes Modellschema den spezifischen Modellcharakeristika des CAD-Modells wesentlich entspricht.

Innerhalb der hier dokumentierten wissenschaftlichen Arbeit wurde eine Kopplung des Modellierers ACIS über ein SDAI an eine Datenbasis realisiert. Die Implementierung des Modellierers ACIS in C++ legt nahe, eine Anbindung an die durch die C++ Spezifikation dieser Schnittstelle definierte und in Kapitel 3.3.7 realisierte Schnittstelle anzugehen.

Eine Realisierung einer Kopplung von ACIS an ein in C realisiertes SDAI wurde innerhalb des EG-ESPRIT-Projektes Prodex vorgenommen. Performanzverluste mit einem Faktor 4 im Vergleich zum lesen eines Modells aus den ACIS-eigenen .sat-Dateien stellen jedoch ein derartiges Konzept in Frage. Der prinzipbedingte Performanzvorteil einer early-binding Implementierung des C++-SDAI soll den Nachteil des Overheads der Datenbankkopplung zumindest teilweise kompensieren.

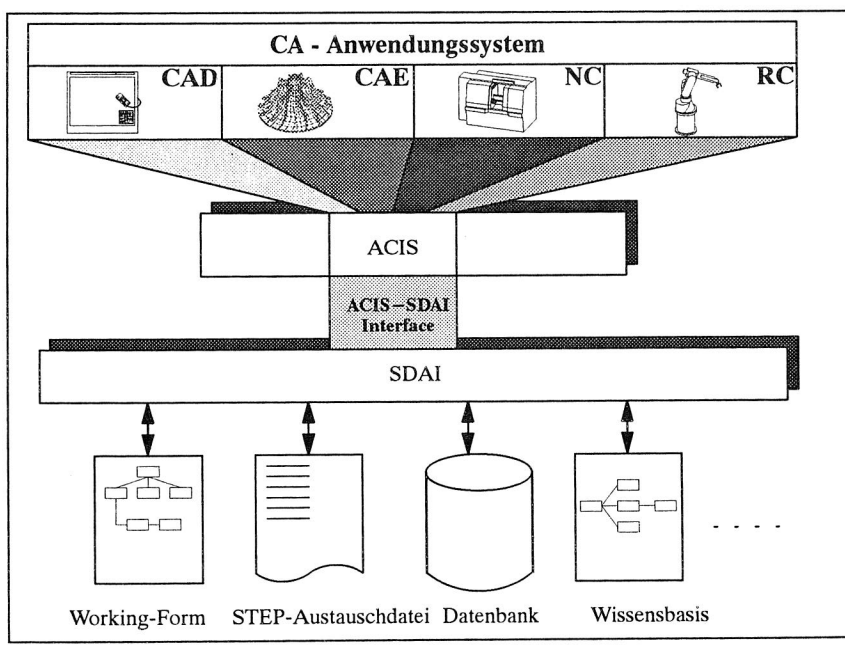


Abb. 38 Kopplung des Geometriemodellierers ACIS an ein SDAI als wesentlicher Teil der Integration technischer EDV in ein Unternehmenskonzept

Das Prinzip der Kopplung des Modellierers ACIS an ein SDAI besteht aus zwei Teilproblemen:

Einerseits der

- Abbildung der ACIS Strukturelemente auf Elemente der STEP Geometrieschemata (und umgekehrt) und
- Erzeugen der SDAI-Instanzen aus ACIS-Datensätzen sowie umgekehrt.

Während erstgenannte Problematik bei der Realisierung des Preprozessors in Kapitel 5.2.3 schon erläutert wurde, muß die zweite Problematik im Folgenden näher erläutert werden.

Wie aus Abb. 38 zu entnehmen, stellt die Kopplung des Modellierers ACIS einen Modell check-in/check-out Mechanismus dar. Dieses Prinzip orientiert sich an der Tatsache, daß einerseits die Einteilung der in einer SDAI-Datenbasis verwalteten Datensätze in sogenannte Repositoriums vorgenommen wird, die etwa einer durch konventionelle Anwendungen erzeugten Dateien entsprechen und andererseits, daß ein Anwendungssystem, das den Modellierer ACIS nutzt, oftmals "hochwertige" Modelle erwartet oder erzeugt. Hochwertig soll in diesem Zusammenhang bedeuten, daß es sich um das topologisch definierte Objekt *'body'* handelt. Das Lesen bzw. Schreiben topologisch niederwertiger Elemente wird daher nur im Kontext eines Gesamtmodellzugriffs vorgenommen.

Eine aus einem ACIS .sat File zu lesende Objektmenge wird immer über das topologisch "höchstwertigste" Element zugänglich gemacht. (Das Isolieren topologisch untergeordneter Elemente ist nach dem Laden natürlich möglich). Im Gegensatz hierzu könnte sich eine über ein SDAI gekoppelte ACIS-Anwendung über einen beliebigen topologischen Einstiegspunkt Zugriff auf unter- (und auch über-) geordnete Elemente verschaffen. Dies würde jedoch in dem auf ACIS basierenden Anwendungssystem komplexe Maßnahmen zur Sicherung der Modellkonsistenz verlangen. Dies soll hier jedoch nicht weiter verfolgt werden.

Der check-in/check-out Mechanismus für den Modellierer ACIS verbirgt vor der Anwendungsprogrammierschnittstelle des ACIS völlig, aus welchem Medium ACIS seine Modelle erhält. Darüberhinaus wird die Problematik der unterschiedlichen Zielvorstellungen der Modellschemata des STEP-Geometrieschemata sowie des ACIS-Modellschemas verborgen. Die wenig anwendungsorientierte Ausprägung des STEP-Modellschemas, das sich z.B. durch fehlende Rückwärtsverweise der einzelnen Modellelemente auszeichnet, die sich jedoch für die Implementierung von Anwendungssystemen als kritisch herausstellt, kann hierbei relativiert werden.

5.3.2 Realisierung der ACIS-SDAI-Kopplung

Die Realisierung der ACIS-SDAI Kopplung folgt einerseits der Spezifikation der C++-Version der in /45/ spezifizierten SDAI-Sprachanbindung; andererseits gibt die in

Kapitel 5.2.3 vorgestellte Abbildung der Geometrieelemente den semantischen Rahmen der Kopplung vor.
Daher soll im Folgenden nur auf die für die speziellen Anforderungen dieser Kopplung nötigen Problemlösungen eingegangen werden.

Während das SDAI als ein API für Anwendungssysteme konzipiert wurde, die die Datenbasis während des Programmablaufes beliebig erfragen können, ist diese Technik für eine Kopplung für einen Modellierer nicht sinnvoll. Die Nutzung von Geometriemodellen innerhalb eines Modellierers (analog einem CAD-System) basiert auf der Verfügbarkeit eines Modells innerhalb des Speicherraums des Modellierers. Zu diesem Zweck besitzen Modellierer die Fähigkeit ein Modell laden zu können. Dieser Ladevorgang muß zum "vollständigen" Laden des Modells führen. Die hier genannte Vollständigkeit bezieht sich auf die topologische Abgeschlossenheit des Modells. Damit muß für die Anbindung des Modellierers ein vollständiges Check-in bzw. Check-out eines Modells vorgenommen werden.

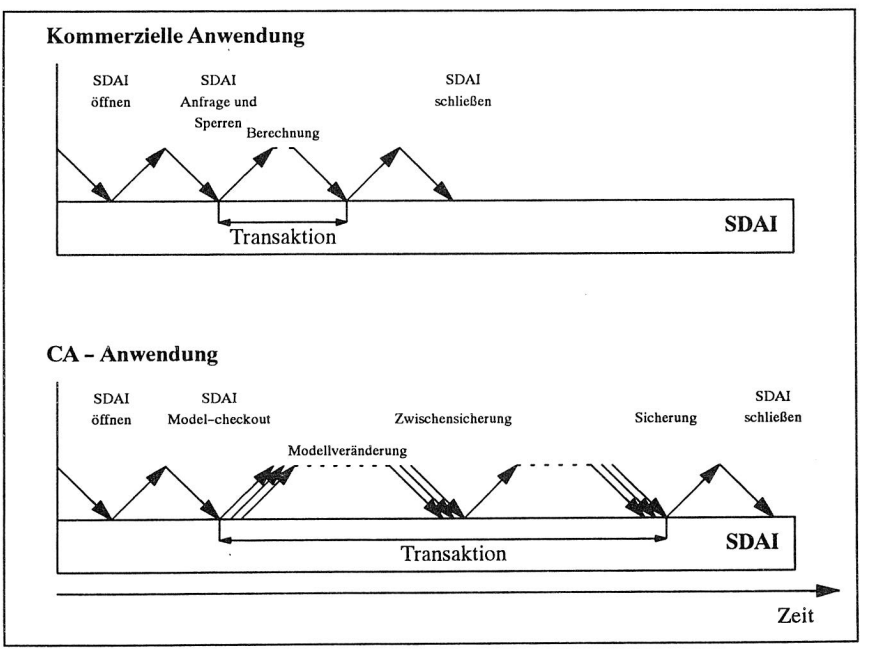


Abb. 39 Problematik der langen Transaktionen bei der ACIS-SDAI-Kopplung

Die Fähigkeit des SDAI ACID-ähnliche Transaktionen zu unterstützen kann damit nur eingeschränkt genutzt werden. Als Transaktion muß damit immer der Zeitraum verstanden werden, der vom öffnen des SDAI bis zum Schließen des SDAI reicht. Intertempo-

räre Zwischenspeicherungen unterbrechen eine Transaktion nicht, sie können allenfalls als definierte Rücksetzpunkte dienen.

Die Abarbeitung aus der Datenbasis über das SDAI auszucheckender STEP Geometrieobjekte greift als Wurzel eine Instanz eines *advanced_brep_shape_representation* (die auch eine Instanz vom Typ *manifold_solid_brep* darstellt) und traversiert die darin vorhandenen Strukturen wie in Kapitel 5.2.3 dargestellt. Die Traversierung des Modellschemas erfolgt im Gegensatz zu der Prozessorimplementierung auf der Möglichkeit, das SDAI über die Struktur eines beliebigen durch das SDAI implementierten EXPRESS-Schemas zu erfragen. Dieses im sogenannten *'Data Dictionary'* abgelegte Metamodell stellt in diesem Fall eine äquivalente Möglichkeit zur Bestimmung des Modellcharakters wieder, wie das in der Prozessorimplementierung benutze Metamodell der Working-Form des Austauschdateiparsers (Kapitel 3.3.2).

Das Einchecken von ACIS-Objekten stellt ein hierzu spiegelbildliches Verfahren dar. Als Ausgangselemente dienen ACIS *'bodies'*, die hinsichtlich ihrer Struktur traversiert werden. Bei der Konstruktion von STEP Instanzen abzubildender ACIS-Elemente war ein Defizit der zu diesem Zeitpunkt aktuellen SDAI-Spezifikation aufgefallen: Im Gegensatz zu der Abbildung von EXPRESS Entitäten auf eine Übertragungsdatei wurde für das SDAI der Konstruktor einer Entity-Instanz mit ererbten Instanzen nicht definiert. Bis dahin hätte die Erzeugung z.B. eines *'cartesian_point'* bestenfalls durch einen Konstruktor, der alle innerhalb des Entity definierten Attribute aufnimmt, und im Nachhinein die Werte der ererbten Attribute gesetzt werden:

Die durch:

```
ENTITY representation_item;
        name : label;
WHERE
        wr1 : SIZEOF(using_representations(SELF)) > 0;
END_ENTITY;

ENTITY cartesian_point
  SUBTYPE OF (point);
        coordinates : LIST [] OF length_measure;
END_ENTITY;
```

definierten Entitäten hätten z.B. in einer komplexen Entityinstanz von Instanzen der Entitäten *representation_item* und *cartesian_point* mit den beiden Funktionsaufrufen:

```
Coordinate_pointH cpt;
cpt = Coordinate_point(1.,2.,3.);// Konstruktor von Coordinate_point
cpt->SetName("point1"); // Setzen des Wertes des ererbten Attributes name
```

vorgenommen werden müssen. Obwohl der Aufbau der Entityinstanz *'cpt'* zum Programmerstellungszeitpunkt bekannt ist, gab es keine Konstruktorfunktion, die durch Angabe aller Attribute der Entität eine Instanz dieser Entität erzeugen hätte können. (In der Version N393 der SDAI-Spezifikation ist eine derartige Konstruktorfunktion aufge-

nommen worden; analog zu der Definition des Mappings für die STEP-Übertragungsdatei werden die ererbten Attribute über die Supertypdefinitionen gesammelt und vor den lokal definierten Attribute in die Parameterliste der Konstruktorfunktion aufgenommen).

6 Anwendungsprotokolle zur Beschreibung von Softwaresystemen

Die in den Kapiteln 3 bis 5 behandelte Problematik richtete sich auf die Realisierung bzw. Implementierung von in EXPRESS definierten Datenmodellen. Dabei wurde auf die spezifischen Inhalte dieser STEP-Datenmodelle (Anwendungsprotokollen) nicht eingegangen. Aus dieser Sicht wurde von den Inhalten eines Datenmodells abstrahiert. In diesem Zusammenhang war zunächst davon ausgegangen worden, daß ein solches Datenmodell für die gewünschte Anwendung bereits existiert. Für die dabei behandelte Problematik ist es aber irrelevant, woher ein solches Datenmodell stammt.
Im folgenden wird nun darauf eingegangen, wie ein solches Datenmodell überhaupt entsteht, wie es entwickelt wird und wie es zu verstehen ist.
Dies ist im Hinblick auf Kapitel 7.2 zu verdeutlichen, wo ein Teil eines solchen Datenmodells entwickelt wird, bzw. eine Anwendung realisiert wird, die auf diesem Anwendungsmodell basiert.

6.1 Anwendungsprotokolle als informationstechnische Grundlage von Anwendungssystemen

Das wesentliche Konzept von STEP ist die Bereitstellung von Anwendungsprotokollen (*application protocols*) als Mechanismus zur Beschreibung von Informationsbedarfen und der Vorlage eines implementierbaren Datenmodells, das diesen identifizierten Informationsbedarf erfüllt /172/.
Ein Anwendungsprotokoll beschreibt den Kontext, den Gültigkeitsbereich und den Informationsbedarf für bestimmte Anwendungen und beschreibt die datentechnischen Konstrukte, die benötigt werden, diesen Bedarf zu erfüllen.

Mit der Orientierung der Spezifikationssprache EXPRESS auf die Beschreibung von Daten, also statischen Sachverhalten, und der zunächst innerhalb der STEP-Entwicklung beschränkten Sicht auf nicht dynamisches Verhalten, sollte in APs nur die Möglichkeit gegeben werden, Informationen zwischen rechnergestützten Anwendungssystemen auszutauschen bzw. für mehrere Anwendungssysteme gemeinsam nutzbar zu machen. Mit den in Kapitel 2.3.3 vorgestellten Erweiterungen von EXPRESS, die prinzipiell alle auf die Erweiterung der Beschreibbarkeit von dynamischen Verhalten hin orientiert sind, soll die ursprüngliche Einschränkung von EXPRESS/STEP auf statische Sachverhalte, aufgehoben werden. Damit würde jedoch eine weit über die Beschreibung von Sachverhalten hinausgehende Festlegung und Normierung vorgenommen werden. Die Standardisierung würde damit einen nicht unwesentlichen Teil von Verhaltensmodellen erfassen, normieren und damit auf eine unabsehbar lange Zeit festlegen. Auch wenn sich argumentieren ließe, das Verhalten von Anwendungssystemen entspricht weitgehend bekannten, seit geraumer Zeit unveränderlichen Methoden, würde hiermit der Weiterentwicklung von Problemlösungen jeglicher Raum genommen. Die Einschränkung auf Normierung von Verhaltensbeschreibung auf kleine Granularitäten ist

hierbei kein treffendes Gegenargument, da sich wohl keine geeignete Grenze für eine akzeptable Granularität angeben läßt. Vor diesem Hintergrund ist auch die Verantwortung der Modellentwickler für die Bereitstellung eines statischen Modells zu sehen.

Zur Zeit der Erstellung diese Schrift sind innerhalb des SC4 die Entwicklung von 32 Anwendungsprotokollen bekannt. Die Auflistung bzw. Beschreibung dieser Anwendungsprotokolle würde einen hier nicht sinnvollen Rahmen annehmen und darüberhinaus nur eine zeitlich limitierte Aussagekraft bedeuten, so daß hierauf verzichtet wird. In einschlägiger Literatur bzw. in Veröffentlichungen des SC4 können die jeweils aktuellen Entwicklungen nachgelesen werden.

Die Implementierungen von STEP basieren auf den Anwendungsprotokollen. D.h. ein Softwaresystem, das eine STEP-Austauschdatei lesen oder schreiben kann oder über die Möglichkeit verfügt, mit Hilfe einer SDAI-Schnittstelle Daten aus einem Datenhaltungssystem zu lesen oder in ein solches zu schreiben, muß dies konform zu einem oder mehreren Anwendungsprotokollen tun.

Im folgenden wird die Entwicklungsmethodik und die daraus entstehenden Probleme von STEP Anwendungsprotokollen beschrieben. Für einige Problematiken sollen Lösungsvorschläge dargestellt werden. Die Darstellung der Entwicklungsmethodik wird für die Einordnung der Erweiterung des Anwendungsprotokolls 210 (Printed Circuit Assembly Product Design Data) zur Beschreibung von Leiterplatten in Kapitel 7 nötig.

6.1.1 Methodik der Entwicklung und Inhalt der Anwendungsprotokolle

Der Aufbau und die Entwicklung von Anwendungsprotokollen ist in /172/ beschrieben. Dieses Dokument ist bisher jedoch nicht Teil der Norm selbst, womit eine direkte Verbindlichkeit nicht automatisch abgeleitet werden kann. Jedoch halten sich die bisher normierten APs an die hier vorgegebene Methodik. Die im folgenden aufgezeigten Probleme führen zu Forderungen, diese Methodik zu ändern, um den AP-Entwicklern für zukünftig zu entwickelnde APs bessere Entwicklungsmethoden an die Hand geben zu können.

Ein Anwendungsprotokoll wird mit Hilfe von drei Modellen beschrieben:

- Das Anwendungsaktivitätenmodell (*application activity model*, AAM)
  Das Anwendungsaktivitätenmodell beschreibt die Aktivitäten und Prozesse, die die Produktdaten in einem spezifischen Anwendungskontext erzeugen und benutzen. Damit soll die Anwendung bzw. die Tätigkeiten beschrieben werden die durch das AP unterstützt werden soll.
  Das Anwendungsaktivitätenmodell wird in IDEF0 beschrieben. IDEF0 ist eine formale, graphische Prozeßbeschreibungssprache. Sie wurde innerhalb des ICAM (Integrated Computer Aided Manufacturing) Projektes von der US Air Force entwik-

kelt und innerhalb der Modellierungstechnik SADT (Structured Analysis and Design Technique) verwendet. Basierend auf hierarchisch und sequentiell strukturierten Diagrammen werden das graphische Symbol Rechteck als Funktion bzw. Tätigkeit und das Symbol Pfeil in Abhängigkeit seiner Anordnung zum Rechteck als Datenflußgrößen, Steuerflußgröße oder Funktionsprinzip zur Modellierung benutzt (Abb. 40).

Obwohl das AAM ein dynamisches Modell zur Beschreibung von Aktivitäten darstellt, die die Verarbeitung von Produktdaten betreffen, ist das Informationsmodell das im AP beschrieben wird, statischer Natur. Dies kann entweder durch die Beschreibung eines Datenmodells erreicht werden, das für alle Zustände gilt, die innerhalb des durch das AAM beschriebene Prozeßmodell möglich sind, oder aber, was durch die bisher bekannten AP deutlich wird, durch die Abstraktion solcher Zustände. Die Erweiterung von EXPRESS um dynamische Sachverhalte (Kapitel 2.3.3) läßt sich zumindest hierdurch motivieren.

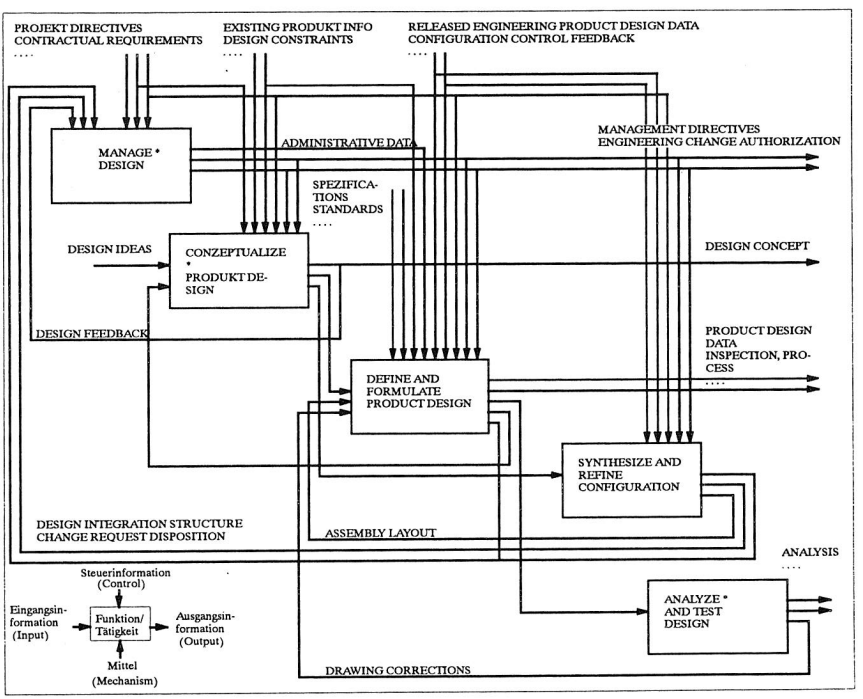


Abb. 40 IDEF0 Diagramm und Anwendungsprinzip (aus /61/)

- Das Anwendungsreferenzmodell (*application reference model*, ARM)
  Das Anwendungsreferenzmodell stellt ein implementierungsunabhängiges Datenmodell dar, das die Daten, die die Anwendung benötigt, in der Terminologie der Anwendung beschreibt. Dieses Modell wird aus dem AAM abgeleitet und formal in ei-

ner oder mehreren Spezifikationssprachen (EXPRESS, EXPRESS-G, IDEF1X oder NIAM) und informell durch die Angabe sogenannter Anwendungsobjekte (*application objects*) sowie deren Beziehungen und Kardinalitäten (*application assertions*) dokumentiert.

- Das anwendungsinterpretierte Modell (*application interpreted model*, AIM)
  Das anwendungsinterpretierte Modell wird aus dem ARM abgeleitet. Die Ableitung (Interpretation) erfolgt durch Identifizierung von Konstrukten aus den Generic Resources, die um spezifische Zwangsbedingungen, in EXPRESS zu definierende lokale und globale Regeln (WHERE Rules, Global RULEs), erweitert werden. Spezialisierungen von Konstrukten aus den 'Generic Resources' sind möglich.

Die Problematik der Beschreibung des ARM in einer der o.g. Modellierungssprachen stellt ein Problem für die weitere Entwicklung dar. Die Modellierungsexperten, die die Überführung der ARM in das AIM vornehmen, müßten in allen o.g. Sprachen ausgebildet sein, darüberhinaus verlangt die 'Mapping Table' eine Darstellung des ARM in EXPRESS. Diese Darstellung des ARM kann nicht automatisiert aus den Darstellungen der anderen Notationen gewonnen werden. Deren Gleichwertigkeit ist zudem noch nicht nachgewiesen.
In /172/ wird zur Validierung des ARM die Implementierung dieses Modells vorgeschlagen, um überprüfen zu können, ob dieses Modell alle Informationsbedarfe der im AAM beschriebenen Anwendung erfüllen kann. Dieser Validierungsmethode stellt einen unzumutbaren Entwicklungsaufwand für dieses Modell dar. Eine Vollständigkeit des Modells scheint derzeit nur informell überprüfbar zu sein.

6.1.2 Die Methodik der Interpretation von Anwendungsreferenzmodellen zu Anwendungsinterpretierten Modellen

Die 'Interpretation' des durch die Anwendungsexperten erstellten Anwendungsreferenzmodell zum anwendungsinterpretierten Modell stellt die wesentliche Aufgabe der Erstellung des anwendungsinterpretierten Modells dar.

Der Weg der Abbildung (Mapping) eines ARM Konstruktes durch ein Konstrukt der Generic Resources wird durch die Abbildungstabelle (Mapping Table) dokumentiert. Für jedes Element des ARM (application object), das in der ersten Spalte der Tabelle genannt wird, wird in der zweiten Spalte der Abbildungstabelle das hierfür identifizierte Element der Generic Resources angegeben. Bei der Definition zusätzlicher globaler Regeln, die für das ausgewählte Element der Generic Resources innerhalb des AIM definiert werden, wird ein Verweis auf diese Regel in der dritten Spalte angegeben. Die vierte Spalte enthält die Quelle, aus dem das identifizierte AIM-Element entnommen wurde. Dies kann eine Generic Resource, ein AIC (s.u.) oder das AIM selbst sein. Die fünfte Spalte enthält bei Bedarf den Referenzpfad (reference path). Mit Hilfe des Referenzpfades wird ein Weg durch den von den Elementen der Generic Resources und der

AICs aufgespannten Modellnetzwerk zu dem identifizierten AIM-Element angegeben (Tab. 7). Dabei ist die Verbindung über Subtypen, Supertypen, Attributen, Auswahltypen oder Aufzählungstypen durch jeweils eigene Verbindungssymbolik angegeben.

APPLICATION ELEMENT	AIM ELEMENT	SOURCE	RULES	REFERENCE PATH
PRODUCT_CONFIGURATION	configuration_item	44	12, 13	
item_id	configuration_item .identification	44		
phase_of_product	configuration_item.purpose	41		
product_configuration to approval	PATH		12	configuration_item approved_item = configuration_item approved_item<- cc_design_approval.items[i] cc_design_approval
product_configuration to part	configuration_design.design	44		configuration_item<- configuration_design.configuration configuration_design configuration_design.design ->product_definition_formation{=>product_definition_formation_with_specified_source} product_definition_formation.of_product->product
product_configuration to planned_effectivity	PATH			configuration_item<- configuration_design.configuration configuration_design<- configuration_effectivity.configuration [configuration_effectivity] [product_definition_effectivity]
PRODUCT_MODEL	product_concept	44	38	

Tab. 7 Abbildungstabelle (Mapping Table) für einen Ausschnitt des AP 203

In /174/ wird der Prozeß der Selektion und Interpretation von Elementen der Generic Resources als die wesentliche Modellierungsproblematik in STEP erkannt. Die beschriebenen Methoden und Lösungsvorschläge bleiben jedoch beliebig und wenig fundiert wie im folgenden gezeigt wird.

Die Auswahl der Konstrukte, durch die ein ARM-Element identifiziert werden soll, unterliegt natürlich den Erfahrungen der Modellierer, darüberhinaus aber auch der Qualität der Definition des ARM. Innerhalb der Abbildungstabelle soll das Ergebnis des Identifikations- und Abbildungsvorganges dokumentiert werden. Die Kritik an der Abbildungstabelle ist jedoch vielfältig:

So ist die Sinnhaftigkeit des Referenzpfades mit der in /172/ angegebenen Spezifikation äußerst zweifelhaft. Zum einen wird für ARM Basiselemente, die auf Entitytypen abgebildet werden, der Supertyp/Subtyp-Pfad dokumentiert, der aber nach gängiger Praxis nicht über alle Supertypen hinweg angegeben wird /61/. Dazu ist diese Information überflüssig, da jeder Entitytyp in seinem Definitionsbereich eindeutig ist und damit der Pfad zu dem identifizierten AIM-Element implizit gegeben ist.

Der in der Spalte der Quelle (Source) angegebene Teil von STEP ist hingegen nicht eindeutig, da innerhalb eines STEP Parts mehrere Schemata definiert sein können, die Entitäten mit identischem Namen vereinbaren; eine Angabe des Quellschemas in Verbindung mit dem STEP-Part würde dieses Identifikationsproblem lösen.

Bei der Bildung des Referenzpfades lassen sich folgende Fälle und Vorgehensweisen unterscheiden /174/:

- Wird das AIM–Zielkonstrukt durch einen Subtyp der Generic Resources beschrieben, so muß der Supertyp aus den Generic Resources für den im AIM neu eingeführten Typ angegeben werden.
- Werden ARM Konstrukte und deren Datenelemente nicht durch **ein** AIM-Entity und dessen **eigene** Attribute interpretiert, so muß für die Verbindung des identifizierten AIM-Entities und dem identifizierten AIM-Attribut ein sogenannter Referenzpfad angegeben werden, der den Pfad vom identifizierten Entity zum identifizierten Attribut angibt.
- Muß einem Attribut des identifizierten AIM-Entity ein spezifischer Wert zugewiesen werden, müssen in dem Referenzpfad die möglichen Werte der Attribute angegeben werden.
- Eine Beziehung zwischen ARM-Konstrukten, die nicht direkt durch Beziehungen der Generic Resources abgebildet werden kann, wird durch einen Referenzpfad angegeben.

Wird für ein ARM-Element ein zugeordnetes Datenelement angegeben, so muß für dessen Mapping ein Referenzpfad angegeben werden, der die Verbindung des identifizierten AIM-Elementes des ARM-Vaterkonstruktes zu dem identifizierten AIM-Elementes des zu mappenden ARM-Elementes angibt (Tab. 7).
Die Tatsache, daß für ein ARM-Element (*application object*) eine bestimmte AIM-Entität und das zu mappende ARM-Datenelement ein Attribut einer anderen AIM-Entität selektiert werden kann, einen Hinweis auf ein fehlerhaftes Modell der Generic Resources oder, wohl häufiger, auf ein vereinfachtes ARM zu geben. Wird ein Konzept der ARM auf ein Konzept des AIM abgebildet und können die Elemente des Ausgangskonzeptes nicht mit Eigenschaften (Attributen) des Zielobjektes identifiziert werden, so scheint das selektierte Zielobjekt nicht optimal ausgewählt worden zu sein. Daß darüberhinaus diese Art der 'Interpretation' überhaupt möglich ist, stellt einen schwerwiegenden Defekt in der Abbildungsmethodik zwischen ARM und AIM dar.
Als Lösungsvorschlag für dieses Problem ließe sich die Bildung eines spezialisierten Subtyps im AIM sowie die zusätzliche Definition von Attributes innerhalb dieses definierten Typs nennen. Es muß jedoch ein Konstrukt des ARM und dessen Datenelemente durch **ein** AIM–Entity **und dessen** Attribute 'interpretiert' werden.

Oftmals sind die Konstrukte bzw. die Beziehungen innerhalb des ARM eine Vereinfachung realer Sachverhalte, die in den Generic Resources wesentlich ausführlicher dargestellt werden. Die 'Interpretation' des ARM durch das AIM stellt dann im wesentlichen eine dokumentierte Korrektur des ARM dar.

Für die Existenz bzw. Kardinalität der Verbindung zwischen dem selektierten AIM-Entity und dem selektierten AIM-Attribut innerhalb des Modellnetzwerkes der Generic Resources lassen sich zusätzlich folgende Anmerkungen machen:

Es existiert *keine* Verbindung:
Nach der gängigen Praxis wird innerhalb des Mappingvorganges davon ausgegangen, daß die anzugebende Verbindung existiert. Diese Annahme ist formal nicht nachzuweisen. Vielmehr läßt sich ein Trivialmodell postulieren, in dem es keine Verbindung zwischen einzelnen Elementen gibt, und dieses Trivialmodell als Zielschema für einen Mappingvorgang heranziehen. Geht man während des Mappingvorganges aber nur von tatsächlich existierenden Verbindungen aus, so wird das Mapping durch das real existierende Modellnetzwerk eingeschränkt. Die Unveränderlichkeit der Generic Resources stellt damit eine unverhältnismäßig hohe Einschränkung in der AIM-Modellbildung dar. Die nicht nachweisbare Existenz einer beliebigen Verbindung zwischen Elementen des Modellnetzwerkes macht die Angabe eines Referenzpfades beliebig.

Es existiert *genau eine* Verbindung:
Existiert zwischen den zu den verbindenden Elementen genau eine Verbindung, so ist die Angabe des Referenzpfades zwar möglich, stellt aber nur eine informelle Angabe dar.

Es existiert *mehr als eine* Verbindung:
In diesem Fall ist die Angabe eines Pfades sinnvoll, da er eine Auswahl aus Alternativen darstellt. Der Hintergrund der Auswahl der gewählten Verbindung ist jedoch aus dem Referenzpfad nicht ersichtlich. Es werden keine Kriterien für die Auswahl aus mehreren Verbindungspfaden angegeben. Vermutlich werden Modellierer den kürzesten Weg zwischen den zu verbindenden Elementen auswählen. Das Dokument das das methodische Vorgehen zur Erstellung der Abbildungstabelle beschreibt, diskutiert diese Problematik jedoch nicht. Die Problematik scheint hierfür noch nicht erkannt worden zu sein.

Die *Spezialisierung* der im AIM neu definierten Entitäten stellt ein weiteres Problem innerhalb der STEP-AIM-Entwicklungsmethodik dar. Die Spezialisierung von Konstrukten, die nur eine partielle Abbildbarkeit auf Konstrukte der Generic Resources besitzen, kann durch zwei Mechanismen vorgenommen werden: Ein von den Generic Resources spezialisiertes Konstrukt kann um abgeleitete Attribute (DERIVE) erweitert werden. Darüberhinaus kann ein Konstrukt durch lokale (WHERE Rules) oder globale (RULE) Regeln spezialisiert werden. Innerhalb der Regeln kann der Typ und der Wert eines Attributes eines Konstruktes eingeschränkt werden. Das Verständnis der Typisierung in EXPRESS geht von der Mengenorientiertheit der Typen aus. Ein Typ ist damit nichts anderes als die Menge aller Werte, die durch diesen Typ bestimmt sind. Die Einschränkung eines Wertes durch explizite Einschränkung seiner Werte und nicht durch Neudefinition eines Typs stellt damit eine Trennung in Typorientiertheit und Wertorientiertheit dar und macht daher keinen Sinn.

Die Einschränkung der Spezialisierungsmaßnahmen innerhalb der STEP-AP-Entwicklungsmethodik stellt eine gravierende Einflußnahme in die Modellbildung des AIM dar und wird durch die weitreichenden Möglichkeiten der Sprache EXPRESS in keiner Weise gerechtfertigt. Modellierungsmethoden, die eine objektorientierte Modellbildung

unterstützen /89/, sehen das Prinzip der Spezialisierung als eines der wesentlichen Modellierungsprinzipien an. Die typisierte Spezialisierung stellt hierbei das wesentliche Modellbildungsprinzip dar. Im weiteren soll versucht werden, die constraintbasierte Modellbildung innerhalb eines STEP-AIM durch die typisierte Spezialisierung zu ersetzen.

Die oben dargestellte Methodik in der Bildung eines AIM stellt eine der schwerwiegendsten Probleme für eine sinnvolle Modellbildung für ein AIM dar. Die typisierte Spezialisierung, die nach der EXPRESS Spezifikation wesentlich leistungsfähiger und verständlicher erreichbar wäre, wird hier auf ein unerträgliches Maß eingeschränkt. Die Folgen für ein implementierbares Modell und ein begründeter, sinnvoller Lösungsvorschlag sollen im folgenden kurz dargestellt werden:

Im STEP Anwendungsprotokoll 203 (und einem später definierten AIC 'advanced brep' /65/) wird ein Konstrukt 'advanced_brep_shape_representation' wie folgt definiert:

```
ENTITY advanced_brep_shape_representation
  SUBTYPE OF (shape_representation);
  WHERE
    wr1: SIZEOF(QUERY ( it <* SELF.items | (NOT (SIZEOF([
              'AIC_ADVANCED_BREP.MANIFOLD_SOLID_BREP',
              'AIC_ADVANCED_BREP.FACETED_BREP',
              'AIC_ADVANCED_BREP.MAPPED_ITEM',
           'AIC_ADVANCED_BREP.AXIS2_PLACEMENT_3D'] * TYPEOF(it)) = 1)) ))
              = 0;
      wr2: SIZEOF(QUERY ( it <* SELF.items | (SIZEOF([
              'AIC_ADVANCED_BREP.MANIFOLD_SOLID_BREP',
              'AIC_ADVANCED_BREP.MAPPED_ITEM'] * TYPEOF(it)) = 1) )) > 0;
      wr3: SIZEOF(QUERY ( msb <* QUERY ( it <* SELF.items | (
              'AIC_ADVANCED_BREP.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) ) | (
              NOT (SIZEOF(QUERY ( csh <* msb_shells(msb,
              'AIC_ADVANCED_BREP') | (NOT (SIZEOF(QUERY ( fcs <* csh\
              connected_face_set.cfs_faces | (NOT (
           'AIC_ADVANCED_BREP.ADVANCED_FACE' IN TYPEOF(fcs))) )) = 0)) ))
              = 0)) )) = 0;
      wr4: SIZEOF(QUERY ( msb <* QUERY ( it <* SELF.items | (
              'AIC_ADVANCED_BREP.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) ) | (
              'AIC_ADVANCED_BREP.ORIENTED_CLOSED_SHELL' IN TYPEOF(msb\
              manifold_solid_brep.outer)) )) = 0;
      wr5: SIZEOF(QUERY ( brv <* QUERY ( it <* SELF.items | (
              'AIC_ADVANCED_BREP.BREP_WITH_VOIDS' IN TYPEOF(it)) ) | (NOT
              (SIZEOF(QUERY ( csh <* brv\brep_with_voids.voids | csh\
              oriented_closed_shell.orientation )) = 0)) )) = 0;
      wr6: SIZEOF(QUERY ( mi <* QUERY ( it <* SELF.items | (
              'AIC_ADVANCED_BREP.MAPPED_ITEM' IN TYPEOF(it)) ) | (NOT (
              'AIC_ADVANCED_BREP.ADVANCED_BREP_SHAPE_REPRESENTATION' IN
          TYPEOF(mi\mapped_item.mapping_source.mapped_representation))) ))
              = 0;
END_ENTITY;
```

Das Konstrukt 'advanced_brep_shape_representation' wird als eine Spezialisierung eines "shape_representation" definiert, wobei zur Spezialisierung sechs lokale Regeln (WHERE Rules) angegeben werden. Die lokalen Regeln schränken die Elemente, die als Attribute des Konstruktes "advanced_brep_shape_representation" verwendet wer-

den können durch lokale Regeln ein. Diese "algorithmisierte" Spezialisierung leidet jedoch unter folgenden Nachteilen:

- Regeln sind zumindest für SDAI-Implementierungen nur explizit Überprüfbar. 'WHERE' Rules werden z.B. in Implementierungen von SDAI-Datenbasen nur auf explizite Anforderung hin überprüft. Implementierungen von Prozessoren, die eine Überprüfung von lokalen und globalen Regeln unterstützen sind bisher nicht bekannt.
- Verletzt ein Datensatz eine der Anforderungen, so ist nur dieser Fakt zu identifizieren, nicht aber den Grund hierfür. Maßnahmen zur Korrektur eines Fehlers sind unbekannt oder sogar unmöglich.
- In der Modellierungspraxis äußerst komplexe Regeln gebildet, die nur noch schwer verständlich sind. Die Korrektheit der beschriebenen Regel ist nur zur Laufzeit einer Implementierung zu überprüfen. Da bisher keine Implementierungen bekannt sind, die die Überprüfung von Regeln unterstützen, kann die Korrektheit der beschriebenen Algorithmen bisher nicht einmal durch Implementierungen validiert werden.
- Regelbasierte Spezialisierungen müssen bei der Verwendung des spezialisierten Konstruktes in anderen Konstrukten wiederholt werden. Diese Art der Spezialisierung unterstützt keine Wiederverwendung.

Im Gegensatz zu der algorithmischen Spezialisierung bietet die Sprachdefinition von EXPRESS prinzipiell die Möglichkeit, die geforderte Konkretisierung eines Entitytyps durch typisierte Spezialisierung der Attribute zu ermöglichen, wobei die o.g. Nachteile ausgeglichen werden können, ohne sich dadurch andere Nachteile zu erkaufen. Eine Neuformulierung des Konstruktes "advanced_brep_shape_representation" könnte dabei wie folgt vorgenommen werden:

Die Regel "wr1" in der obigen Definition besagt, daß als Typ des ererbten Attributes "items" des Entity "advanced_brep_shape_representation" nur Objekte vom Entitytyp "manifold_solid_brep", faceted_brep, "mapped_item" oder "axis2_placement_3d" zuzulassen sind.

Die Neuformulierung der Spezialisierung die durch die lokale Regel "wr1" ausgedrückt werden soll, könnte durch die Einführung eines neuen SELECT Typs und durch Spezialisierung des Attributes items auf diesen Auswahltyp erreicht werden. Die lokale Regel "wr1" wird damit überflüssig:

```
TYPE absr_basetype = SELECT (
  manifold_solid_brep,
  faceted_brep,
  mapped_item,
  axis2_placement_3d
);
END_TYPE;
```

```
  ENTITY advanced_brep_shape_representation
    SUBTYPE OF (shape_representation);
  SELF\representation.items : SET [1:?] OF absr_basetype;
    WHERE ...
```

Die Regel wr3 fordert darüberhinaus, daß alle Flächen, die in den "Shells" eines manifold_solid_brep verwendet werden, vom Typ "advanced_face" sind. Diese Forderung ließe sich weiterhin durch eine typisierte Spezialisierung des manifold_solid_brep ausdrücken. Hiermit läßt sich "wr3" eliminieren:

```
TYPE absr_basetype = SELECT (
  af_manifold_solid_brep,
  faceted_brep,
  mapped_item,
  axis2_placement_3d
);
END_TYPE;

ENTITY af_closed_shell
  SUBTYPE OF (closed_shell);
  SELF\connected_face_set.cfs_faces : SET [1:?] OF advanced_face;
END_ENTITY;

ENTITY af_manifold_solid_brep
  SUBTYPE OF (manifold_solid_brep);
  SELF\manifold_solid_brep.outer : af_closed_shell;
END_ENTITY;

  ENTITY advanced_brep_shape_representation
    SUBTYPE OF (shape_representation);
  SELF\representation.items : SET [1:?] OF absr_basetype;
    WHERE ...
```

Ebenso wie die gezeigten Veränderungen des Ausgangsschemas läßt sich das Objekt "advanced_brep_shape_representation" durch Definition spezialisierter Subtypen von Objekten der Generic Resources oder der AICs ausdrücken (af_closed_shell, af_manifold_solid_brep). Die dadurch ausgedrückte Spezialisierung der jeweiligen Objekte ist weitaus verständlicher und verhält sich dabei konservativ, d.h. sie erfordert prinzipiell keine Erweiterung von Softwaresystemen, die bereits in der Lage sind, Schemata zu verarbeiten, die nur auf der Verwendung von Konstrukten der Generic Resources basieren. Die Definition eines neuen SELECT Typs (absr_basetype) sollte eine Anwendbarkeit für bestehende Softwaresysteme dann nicht behindern, wenn diese Softwaresysteme in der Lage sind, SELECT Typen über deren Basiselemente zu verarbeiten. Die Bildung neuer SELECT Typen ist dabei durch die AIM-Entwicklungsmethodik gedeckt.

Eine weitere Problematik der AP-Entwicklung wird durch die Art der Dokumentation des entstandenen AIM hervorgerufen. Das durch den Mapping-Vorgang entstandene anwendungsinterpretierte Modell wird in in einem sogenannten short-form Schema dokumentiert. In einem short-form Schema werden die aus den Generic Resources, den Integrated Resources und den AICs verwendeten Konstrukte durch Interface-Definitionen

(USE FROM, REFERENCE FROM) oder implizit eingefügt. Darüberhinaus wird hieraus ein long-form Schema generiert. Diskrepanzen zwischen beiden Schemata sollen dabei nicht auftreten.
Auch diese Forderung ignoriert einige der Probleme die hierbei auftreten können.

Durch das Einfügen von Konstrukten die zur Vollständig- und Verständlichkeit andere Konstrukte benötigen, werden Konstrukte in das long-form-Schema übernommen, die innerhalb des short-form-Schemas und damit im AIM des Anwendungsprotokoll nicht definiert sind. Bei Generierung eines korrekten long-form Schemas existieren im long-form Schema prinzipiell (im Sinne von EXPRESS) instantiierbare Entitäten, obwohl sie im short-form Schema nicht genannt werden.
Beispiel hierfür ist eine für ein korrektes EXPRESS Schema fehlende Definition der Entitäten "derived_unit" und "derived_unit_element" in der IS-Version des long-form Schemas zur Beschreibung von Zeichnungen /60/. Die o.g. Elemente werden von der im Schema benötigten Funktion "derive_dimensional_exponents" referenziert. Nach Einfügen dieser Elemente in das long-form-Schema sind Instanzen dieser Entitäten in einem AP201 Datensatz prinzipiell (nach EXPRESS-Sprachdefinition) möglich, auch wenn sie innerhalb des Anwendungsprotokolles nicht definiert werden.

6.1.3 Nutzung von Anwendungsprotokollen in technischen Anwendungssystemen

Die Anwendungsprotokolle in STEP definieren ein Informationsmodell in einen Anwendungskontext, das es ermöglichen soll alle innerhalb dieses Anwendungskontextes möglichen Informationen durch Daten zu beschreiben um damit einen Datenaustausch zwischen Softwaresystemen oder Datenbasen für den Anwendungsbereich zu ermöglichen. Die Definition des Informationsmodells aus Anwendungs- oder Anwendersicht wird innerhalb des Anwendungsprotokolls durch das ARM beschrieben. Für ein Anwendungssystem ist daher die Möglichkeit gegeben Daten, die mit Hilfe des Anwendungsreferenzmodells implementiert wurden, eine Sicht des Anwendungssystems auf die durch die Implementierung ermöglichten Daten zu erreichen.

Ein Anwendungssystem könnte damit wie in Abb. 41 skizziert implementiert werden: Das Anwendungssystem bietet dem Anwender die Erzeugung, Veränderung, Löschung von Datenobjekten in der dem Anwender vertrauten Sprache an, wie sie innerhalb des ARM definiert wurde. Mit Hilfe der Mapping Table muß das Anwendungssystem die Daten nun in die von AIM definierte Repräsentation bringen, um sie einer STEP-Implementierung zuführen zu können. Andererseits muß bei Bedarf das Anwendungssystem in der Lage sein, Daten von der Implementierung, dargestellt durch das AIM, in die Repräsentation des ARM zurückführen zu können. Dies muß über die Invertierung der Abbildungsvorschrift der Mapping Table geschehen.

Die hierbei aufgeworfene Frage lautet also:

Gibt es eine Invertierbarkeit der Mapping Table, d.h. können Daten, die im Format des AIM vorliegen **eindeutig** in das ARM umgewandelt werden?

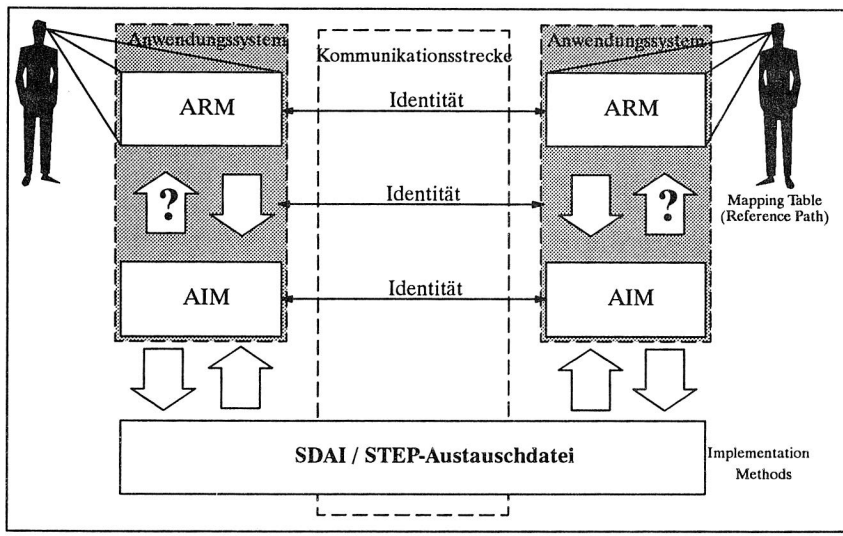


Abb. 41 Implementierung eines Anwendungssystem mit Hilfe des Anwendungsreferenzmodells (ARM) über einem Implementierungsmodell

Diese Problematik läßt sich mathematisch korrekt wie folgt formulieren (Dabei wird eine formale Repräsentation des ARM etwa in EXPRESS vorausgesetzt):
Gegeben sei eine Menge von Werten i (Instanzen) I_{ARM} in der Repräsentation des ARM und eine Abbildungsvorschrift Map.

Es gilt: $I_{AIM} = Map(I_{ARM})$
(mit I_{AIM} als der Menge der Instanzen in der Repräsentation des AIM)
Gesucht wird eine Funktion Map_{inv} so daß gilt:
$Map_{inv}(Map(I_{ARM})) = I_{ARM}$, für alle I_{ARM}

In vielen Veröffentlichungen (z.B. /171/), wird vermutet, eine derartige Funktion Map_{inv} würde nicht existieren. Damit wird jedoch die Existenz der Abbildungsfunktion in Frage gestellt, sondern die Eigenschaft der Abbildung als Funktion hinterfragt. Es ist zu Vermuten, daß diese Abbildung eine im mathematischen Sinne definierte Relation darstellt, so daß für einen unabhängigen Wert (hier eine AIM-Instanz) mehrere abhängige Werte (hier ARM-Instanzen) existieren. Die Analyse der Abbildungsfunktion erscheint manuell nicht durchführbar, da Kontext- und Constraintsabhängigkeiten der Ausgangs- sowie der Zielwerte berücksichtigt werden müssen.

Hingegen wird in /169/ der EXPRESS-Dialekt EXPRESS-V benutzt, um genau eine wie in Abb. 41 dargestellte Sicht auf die durch das AIM implementierten Daten zu erzeugen.

Würde eine funktionale Abbildung Map_{inv} nicht existieren, so wären alle Daten, die in einem System vorrätig wären, das ein AIM implementiert für ein Anwendungssystem völlig nutzlos, sie wären für das Anwendungssystem nicht eindeutig (re-)interpretierbar! Würde hingegen ein Anwendungssystem seine Daten einem Anwender nicht nach einem ARM sondern nach einem AIM darbieten, so ist die Definition eines Anwendungsdatenmodells durch das ARM sinnlos. Die durch ANSI/Sparc definierte und in STEP wiederzuerkennenden Modellierungs- und Implementierungsmethodik geht jedoch auch von einer Implementierung des Anwenderdatenmodells in einem Anwendungssystem aus.

Die Suche nach der Invertierung der Abbildungstabelle (Mapping Table) wird unter anderem auch durch die wenig formale Definition der Abbildungstabelle erschwert. Die Spezifikation des Anwendungsreferenzmodells in einer formalen Darstellung (wie sie in EXPRESS möglich, aber in STEP nicht zwingend vorgeschrieben ist) ist hierfür fundamentale Voraussetzung.

Die Schwierigkeit die bei der Abbildung eines Anwendungsreferenzmodells zu einem anwendungsinterpretierten Modell auftauchen sind offensichtlich auch in der Vermischung der Konzepte des Anwendungsreferenzmodells und des AIM sowie in der Mißverständlichkeit des Begriffs des Anwendungssystems zu suchen (siehe hierzu auch Abb. 42). Es stellt sich z.B. die Frage, ob Anwender im Bereich des Automobilbaues innerhalb ihres Anwendungskontextes den Begriff "Advanced Brep" kennen oder kennen sollten.

6.2 Zur Interoperationsproblematik der Anwendungsprotokolle

Im Rahmen der STEP-Entwicklungsmethodik wurde zu Beginn ein einziges umfassendes Produktmodell angedacht (**I**ntegrated **P**roduct **I**nformation **M**odel, IPIM) /4/. Die Komplexität, die ein solches Modell mit sich bringen würde, wurde offensichtlich zunächst unterschätzt. Seit der ISO-Sitzung 1989 wurde dieses Konzept aber fallengelassen und die Entwicklung eigenständiger, abgeschlossener Informationsmodelle, der Anwendungsprotokolle, durchgesetzt.

Bei Nachweis eines begründeten Bedarfs für ein Informationsmodell kann jeder Interessent ein Informationsmodell entwickeln und dieses der ISO zur Normung vorschlagen. Zum aktuellen Zeitpunkt sind etwa 32 solcher Anwendungsmodelle in Entwicklung bzw. bereits entwickelt und normiert. Diese Informationsmodelle werden jeweils im Hinblick auf die speziellen Anforderungen der Interessenten entwickelt. Die Anforderungen der einzelnen Informationsmodelle werden jedoch nicht orthogonal definiert, so daß die entstandenen Modelle Überschneidungsbereiche besitzen. In Abb. 42 wurde versucht, aus einer Teilmenge der existierenden bzw. in Entwicklung befindlichen Informationsmodelle ein Anwendungsportfolio zu skizzieren.

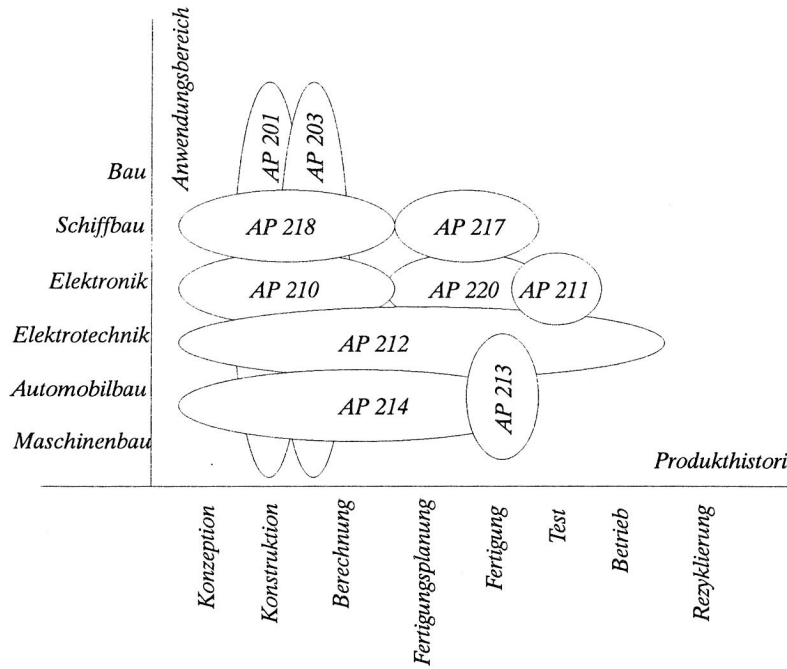


Abb. 42 Gültigkeitsbereiche der bekannten Anwendungsprotokolle

Während Anwendungsprotokolle wie das AP 201 /60/ und das AP 203 /61/ wenig auf industrielle Branchen sondern auf Anwendungsbereiche hin orientiert sind, wurde Anwendungsprotokolle wie das AP 210, das AP 212 und das AP 214 für jeweils spezifische Industriebereiche entwickelt. Da offensichtlich eine Überlagerung der Anwendungsbereiche von APs auftritt, ist zu erkennen, daß sich einerseits die Informationsbedarfe abgegrenzter Problembereiche durch mehrere Anwendungsprotokolle abdecken lassen. Dies gilt z.B. für Technische Zeichnungen im Automobilbau durch AP201, AP202 und AP214. Andererseits aber lassen sich andere Problemstellungen aber (bisher) nur durch die Nutzung mehrerer Anwendungsprotokolle gleichzeitig lösen, wie dies z.B. beim Einsatz von Elektronik oder Verkabelung (AP 210, AP 212, AP 214) im Automobilbau nötig sein wird.

Die Notwendigkeit der Nutzung mehrerer Anwendungsprotokolle für einen Anwendungsfall und die daraus entstehenden Probleme wurden als eine der wesentlichen Problematik innerhalb der STEP-Entwicklung erkannt. Unter der Bezeichnung Interoperabilität (interoperability) wurde hieraus eine eigene Begriffswelt. Zunächst gab es um die Begrifflichkeit hierzu Bedeutungsverschiebungen, da dieser Begriff auch für das Prinzip von STEP verwendet wurde, vorhandene Normen zu vereinnahmen und in STEP zu integrieren. Dies gilt z.B für ANSI-Normen bei Programmiersprachen, IEC-Normen in

der Elektronik, SGML und EDIFACT. Hierfür wurde später jedoch der Begriff der "Kooperativen Nutzung von Standards" (Cooperative Use of Standards) benutzt.

Interoperabilität wird in der Literatur u.a. als die Fähigkeit eines Softwaresystems verstanden, Daten zu verarbeiten und zu nutzen, die innerhalb eines Anwendungsprotokolls A definiert wurden obwohl das Softwaresystem für AP B implementiert wurde.
In /88/ wird hingegen Interoperabilität als die Fähigkeit definiert, Datenmengen durch externe Referenzen von STEP-Austauschdateien zwischen Implementierungen verschiedener APs auszutauschen.

6.2.1 Interoperabilität von gekoppelten System beim Datenaustausch

Für verschiedene Problematiken der Interoperabilität beim Austausch von Datensätzen über STEP-Austauschdateien lassen sich unterschiedliche Szenarien unterscheiden. Für alle Fälle ist angenommen, ein Anwendungssystem ist für den Zeitraum des Lesens oder Schreibens eines Datensatzes konform zu genau einem Anwendungsprotokoll:[16]

- Eine Menge von Datenelementen innerhalb eines Datensatzes, die einem Überlappungsbereich mehrerer APs entsprechen, sollten von allen Systemen die eines der Überlappenden APs implementieren, gelesen und interpretiert werden können.

Der Begriff des 'interpretieren' in den obigen Aussagen stellt die semantische Verarbeitbarkeit der übertragenen Daten in den Vordergrund. Die Fähigkeit, Datensätze korrekt einzulesen und ihre Typkorrektheit und deren Konformität zu einem EXPRESS-AP-Schema zu überprüfen, genügt hierfür nicht.

Dies ist der klassische Fall, wenn zwei Anwendungssysteme unterschiedliche APs als Prozessoren benutzen, obwohl offensichtlich (aus Sicht des ARM) deutliche Überlappungen in den Anwendungsreferenzmodellen bestehen. Realistisch ist dies der Fall, wenn z.B. unterschiedliche CAD-Systeme einerseits einen Prozessor mit Hilfe des AP203 und andererseits mit Hilfe des AP214 anbieten, diese Systeme aber miteinander kommunizieren müssen.
Zu diesem Fall muß eine Unterscheidung in die Fähigkeit eines Anwendungssystems vorgenommen werden, Datensätze lesen zu können, die zu einem Schema konform sind, das nicht dem implementierten Schema entspricht, und der Fähigkeit diese gelesenen Daten sinnvoll in eine Anwendung aufnehmen zu können.

Implementiert im Gegensatz hierzu ein Anwendungssystem strikt die Fähigkeit, ausschließlich die Instanzen lesen und interpretieren zu können, die innerhalb des AIM-Schemas eines Anwendungsprotokolls definiert sind, so muß durch die explizite Erweiterung des Softwaresystems auf die Verarbeitbarkeit zusätzlich der Instanzen vorge-

16. Daß ein Anwendungssystem mehrere APs implementieren kann und bei einer Sequenz von Lese-Schreibvorgängen unterschiedliche AP unterstützen kann, widerspricht dem nicht.

nommen werden, die innerhalb der von Interesse stehenden Anwendungsprotokolle definiert sind.
Nach dem Konzept, das der Implementierung des in Kapitel 3.2.2 vorgestellten Softwaresystems zugrunde liegt, ist die Fähigkeit Datensätze zumindest lesen zu können, die zu einem bestimmten Schema konform sind, nur vom Zugriff auf eine Datei abhängig, die das definierende Schema enthält.
Die Forderung, die innerhalb eines Datensatzes enthaltenen Instanzen interpretieren zu können stellt dagegen wesentlich anders geartete Anforderungen an ein Softwaresystem. Hierbei muß das Softwaresystem über die Aufnahme der Datensätze hinaus diese in ihrem Sinnzusammenhang, wie er z.B. durch das ARM definiert wird, erfassen können und dem Anwender in einer geeigneten Weise darbieten.

- Mehrere Datensätze, die jeweils zu unterschiedlichen APs konform sind, enthalten eine Menge an Informationen, die über einen Teilbereich identisch sein können. Diese müssen von einem Anwendungssystem durch Implementierungen unterschiedlicher APs verarbeitet und über einen in den APs vorhandenen oder zu definierenden Zusammenhang semantisch verbunden werden.

Dies kann notwendig sein, wenn eine Problematik ein Anforderungsprofil besitzt, das vollständig durch kein existierendes AP, aber teilweise durch Teilbereiche vorhandener AP abgedeckt wird, wie dies in /175/ beschrieben wird.

Die prinzipielle Lösung für dieses Problem liegt in der Aufforderung, ein AP für genau den geforderten Anwendungsfall bereitzustellen. Da die AP-Entwicklungsmethodik jedoch wie in Kapitel 6.1.1 erläutert keine effiziente Möglichkeit vorsieht, Module aus anderen AP-Entwicklungen zu übernehmen, stellt die AP-Entwicklung, wie in Kapitel 6.1.1 erläutert, ein äußerst aufwendiges Verfahren dar, so daß nicht für jedes Anwendungsproblem ein "maßgeschneidertes" Anwendungsprotokoll zur Verfügung stehen wird, obwohl alle wesentlichen Teilbereiche des Problems in mehreren unterschiedlichen Anwendungsprotokollen bereits gelöst sind.

6.2.2 Interoperabilität von Anwendungssystemen über SDAI-Datenbasen

Die Problematik der Interoperabilität von Anwendungssystemen, die über Datenbasen Daten nutzen können, die verschiedenen Anwendungsprotokollen zugeordnet werden können, wird für die Implementierung des SDAI weitaus besser unterstützt als dies für den Datenaustausch der Fall ist.

In /120/ wird ein Data Dictionary für SDAI-Datenbasen beschrieben, das jede SDAI-Implementierung bereitstellen muß, um Anwendungssystemen die Möglichkeit zu verschaffen, normierte Anfragen über den strukturellen Aufbau der Datenbasis abzusetzen, und damit eine begrenzte Implementierungsflexibilität gegenüber der angebundenen Datenbasis zu ermöglichen.

In diesem Data Dictionary wird eine Struktur definiert, die vereinbart, welche Instanzen in mehreren, über das SDAI bereitgestellten Anwendungsprotokolle identisch sein können.
Die hierfür wichtigsten in EXPRESS definierte Strukturen aus /44/ lauten hierfür:

```
ENTIYT schema_instance;
name              : STRING;
contents          : SET [0:?] OF sdai_model;
native_schema     : schema_definition;
repository        : sdai_repository;
change_date       : STRING (16) FIXED;
validation_date   : STRING (16) FIXED;
validation_level  : INTEGER;
UNIQUE
url               : name, repository;
END_ENTITY;


ENTITY schema_definition;
name : express_id;
INVERSE
entities          : SET [0:?] OF entity_definition FOR parent_schema;
types             : SET [0:?] OF defined_type FOR parent_schema;
global_rules      : SET [0:?] OF global_rule FOR parent_schema;
external_schemas  : SET [0:?] OF external_schema FOR native_schema;
END_ENTITY;

ENTITY external_schema;
external_schema   : express_id;
for_types         : SET [1:?] OF domain_equivalent_type;
native_schema     : schema_definition;
END_ENTITY;
```

Durch eine 'Schemainstanz' (*schema_instance*) wird eine Menge von Sdai-Modellen (*contents*), die einem spezifischen Schema (*native_schema*) entsprechen, einem Repositorium (*repository*) zugeordnet. Für jede Schemadefinition (*schema_definition*) werden u.a. die innerhalb dieses Schemas referenzierten Schemas (external_schemas) identifiziert.[17]

Hiermit wird ein in Abb. 43 Konzept verteilter Datensätze möglich, die sich einerseits den AIM-Schemas von Anwendungsprotokollen und andererseits der Nutzung verschiedener Repositorien zunutze macht. SDAI-Modelle die sich z.B. den AICs zuordnen lassen, können damit von den AIMs verschiedener APs referenziert werden. Dies kann prinzipiell auch für einen AIC-spezifischen Teil eines Gesamtdatensatzes gelten. Problematisch dabei ist allerdings, daß diese verteilte Nutzung solcher 'Teildatensätze' nicht implizit möglich ist, sondern von einem Anwendungssystem durch explizites Verwalten mehrerer SDAI-Modelle erreicht werden muß. Daß dieses Konzept nur durch Realisie-

17. *Strenggenommen wird in einer EXPRESS-Schemadefinition ein externes Schema nicht explizit, sondern nur implizit über die aus dem externen Schema referenzierten Elemente referenziert. Auch hierdurch wird deutlich, daß es sich bei den SDAI-Modellen um eine Art evaluierter Metamodelle handelt.*

rung des SDAI sowie der Anwendungssysteme mit Hilfe der 'short form Schemas' möglich stellt eine weitere Anforderung dar, die in den bisherigen Implementierungsversuchen wenig Beachtung fand.

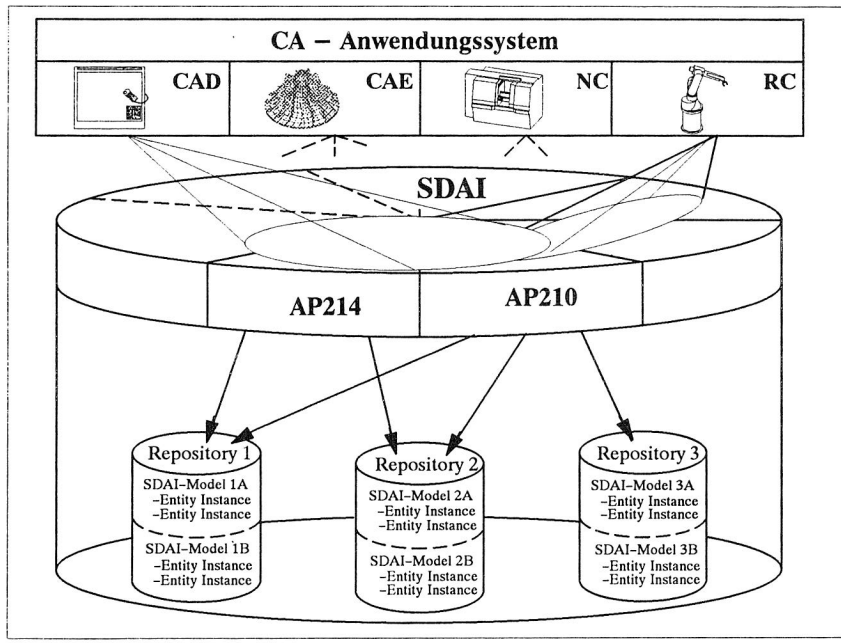


Abb. 43 'Data Sharing' von Anwendungssystemen über SDAI-Datenbasen

Mit diesem Prinzip läßt sich zumindest auf AIM-Ebene die implementierungsorientierte Unterstützung der Interoperabilität verschiedener Anwendungsprotokolle erreichen. Ob eine semantische Interoperabilität von Anwendungen dies überhaupt zuläßt, wurde bereits diskutiert. Eine Lösung dieses Problems muß u.a. auch bei der Entwicklung der APs bereits hinreichend berücksichtigt werden.

6.3 Das Prinzip der Modularisierung von Anwendungsprotokollen

Das Prinzip der Modularisierung ist eine anerkannte Problemlösungsstrategie, die besonders bei der Behandlung komplexer Probleme angewandt wird. Innerhalb der Softwaretechnologie ist die Orientierung auf prozedurale Zerlegung innerhalb prozeduraler Sprachen, auf Modularisierung wie sie von Wirth in Modula favorisiert wurde, aber auch die Objektorientierung /89/ ein Verfahren der Modularisierung.

Für STEP wurde also aufgrund unüberwindlicher Schwierigkeiten bei der Gesamtmodellerstellung eine modulare Entwicklung akzeptiert, die sich wie folgt in einzelnen Modulebenen ausdrückt:
Aufbauend auf den Generic Resources werden die Application Resources entwickelt (siehe Abb. 44). Diese Modularisierung wurde später durch die Einbeziehung der Application Interpreted Constructs (AIC) erweitert. Zunächst sollten diese als Partialmodelle nicht direkt Teil der Norm ISO-10303 sein, dies wurde jedoch als ein formales Problem erkannt. Seit 1995 sind AICs als Serie 500 Teil der Norm.

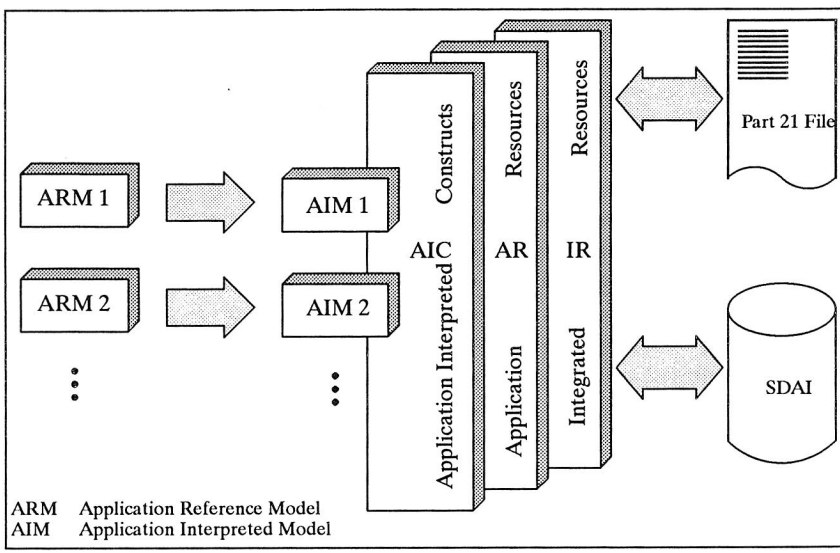


Abb. 44 Mechanismus der Modellbildung von Anwendungsprotokollen

Während Generic Resources, Application Resources und Application Interpreted Constructs eine Modularisierung der Modellbildung im Rahmen der AIM-Definition darstellen, stellt sich auch für die Entwicklung des Anwendungsreferenzmodells die Problematik der Modularisierung. Auf der Ebene des ARM können zur Definition eines funktionalen Zusammenhanges sogenannte Funktionaleinheiten (Units of Functionality, UoF) definiert werden, innerhalb derer die Anwendungsobjekte gruppiert werden. Für jedes AP werden weiterhin sogenannte Konformitätsklassen (Conformance Classes, CC) definiert. Diese stellen jeweils eine Menge von Konstrukten dar, die für die Einhaltung einer bestimmten Konformitätsklasse von einer Implementierung zu unterstützen sind.

Die wesentlichen Fragen bei der Analyse der Modulkonzepte für die Entwicklung von APs lassen sich wie folgt formulieren:

- Können Module (in anderen Anwendungsprotokollen) wiederverwendet werden?

- Sind die Modulkonzepte orthogonal?
- Sind die Modulkonzepte rekursiv anwendbar?
- Bleiben Module entlang der Entwicklung erhalten?

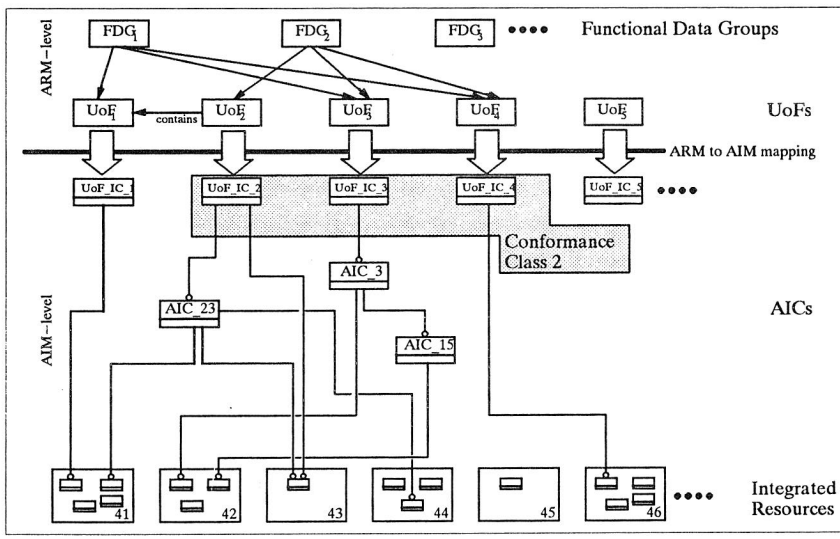


Abb. 45 Modularisierungskonzepte in STEP (nach /171/)

Die wesentlichen der oben gestellten Fragen sollen im weiteren kurz diskutiert, aufgeworfene Probleme dargestellt und Lösungsvorschläge gemacht werden

6.3.1 Modularisierung der Anwendungsinterpretierten Modelle

Die innerhalb der STEP-Methodik bereits definierten und etablierten Partialmodelle, wie sie in den Generic Resources, den Application Resources und den AICs festgeschrieben wurden, stellen eine Modularisierung für die Bildung von AIM Modellen bereit. Dies bedeutet, daß dieses Modularisierungskonzept nur innerhalb der AIM-Modellbildung zum Tragen kommt. Weiterhin ist in STEP die Bildung von Modulen, die über die bereits existierenden Modulebenen hinausgehen, nicht möglich. Alle Anwendungsprotokolle müssen sich mit der Verwendung der in Abb. 46 skizzierten Modulebenen begnügen.

Werden während der AP-Entwicklung Überlappungen zwischen Anwendungsprotokollen festgestellt, so sollen für diese Überlappungsbereiche AICs definiert werden /174/. Die Identifikationsmöglichkeiten für Überlappungen sind dabei äußerst vage. Erst während der Interpretationsvorganges kann erkannt werden, ob Teile eines Anwendungsreferenzmodells auf ein AIC abgebildet werden können. Eine eindeutige Methode zur

Identifikation von Teilen eines ARM auf ein AIC kann es wegen möglicher unterschiedlicher Terminologie nicht geben.

Die Bildung der o.g. Konformitätsklassen sind eine Hilfe für Systemanbieter bei der Entwicklung normkonformer Softwaresysteme. Trotz der Implementierung nur einer Teilmenge aller in einem AP definierten Konstrukte kann ein Hersteller bereits Normkonformität erreichen. So soll ein schrittweiser Einstieg in die Unterstützung der Norm ISO-10303 möglich sein. Die Modularisierung der Konformitätsklassen ist also orthogonal zu den Modularisierungsprinzipien der Modellbildung innerhalb der AP-Entwicklung.

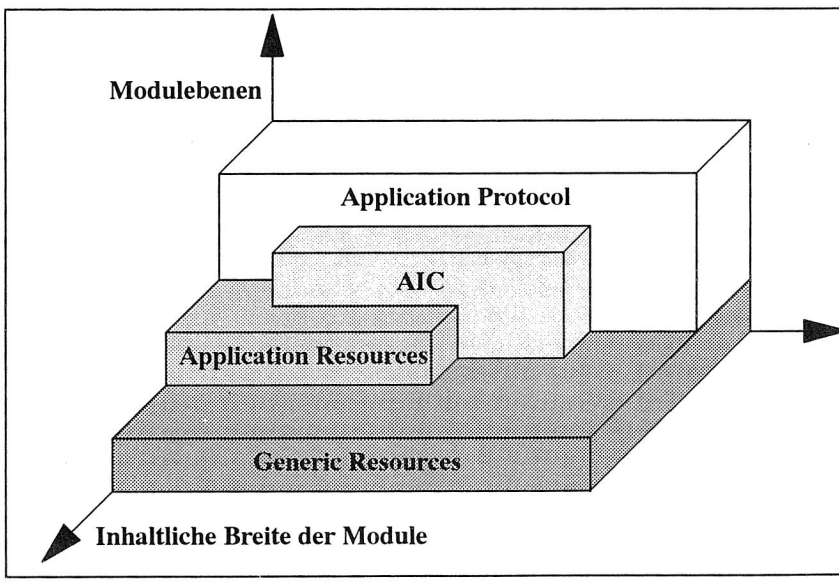


Abb. 46 Modularisierungsebenen von STEP-AIM-Modellen durch die gegebene Modulstruktur

Ein weiterer Grund zur Bildung von AICs soll auch die Unterstützung der Entwicklung von Softwaremodulen sein, die jeweils ein AIC implementieren. Die Bildung der AICs soll parallel der Definition der Konformitätsklassen verlaufen, so daß es möglich sein sollte, mit Softwaremodulen, die jeweils ein AIC realisieren, schrittweise Softwaresysteme zur Unterstützung von APs der ISO-10303 aufzubauen. Allerdings ist die Definition der CCs nicht hierarchisch aufgebaut. Eine Implementierung, die Konform zu CC6 ist, unterstützt nicht notwendigerweise CC2.

Eine Abbildung von Modulen innerhalb des ARM, wie sie durch die Bildung der Funktionaleinheiten (UoF) möglich sind und in /171/ (Abb. 45) gewünscht werden, auf das AIM, wird nicht unterstützt. Mit Hilfe der in /171/ dokumentierten Functional Data Groups

(FDG) soll genau diese Modularisierung und deren Unterstützung über die Entwicklung der APs hinweg, ermöglicht werden. Defizite in der formalen Beschreibung des ARM und der Methodik zur Bildung der FDG verhindern bisher deren Umsetzung.

6.3.2 Alternative Modularisierungsprinzipien

Für die im vorigen Abschnitt dargestellte Problematik können verschiedene Lösungsvorschläge angegeben werden. Dabei muß jedoch beachtet werden, daß diese Vorschläge mit den in der STEP-Entwicklung vorgegeben Methoden nicht konform sind und damit zumindest eine Unterstützung des SC4 für Entwicklungen nach diesen Methoden nicht wahrscheinlich ist.

In /134/ wird für eine Reihe von Anwendungsprotokollen, die einen überschneidenden Anwendungsbereich (Schiffbau) besitzen, eine gegenseitige Referenzierung gemeinsamer Konstrukte vorgestellt. Damit wird die in Abb. 46 dargestellte, abgeschlossene Modularisierungshierarchie derart erweitert, daß Anwendungsprotokolle Teile ihres Interessensbereiches mit anderen Anwendungsprotokollen teilen können.
Methodisch wurde dabei vorgesehen, ein Basis-Anwendungsprotokoll zu definieren, das eine Fundament für darauf aufbauende spezialisierte Anwendungsprotokolle darstellen sollte (Building Blocks).

Problematisch erscheint dabei jedoch die durch die EXPRESS Sprachdefinition schwierige Definition gemeinsam nutzbarer, komplexer Einheiten. Zudem scheint das Konzept auf der Definition der sprachlichen Definition der Zusammenhänge stehengeblieben zu sein. Die damit erreichte Modularisierungsmethodik erweitert das existierende Modularisierungsprinzip nicht wesentlich. Vielmehr erscheint dieses Prinzip die Eigenschaften der bisherigen Module nur zu verschieben.

Alternativen zur Lösung der Interoperationsproblematik zwischen unterschiedlichen Anwendungsproblematik wäre die vollständige Integration von Anwendungsprotokollen in andere Anwendungsprotokolle. Während einerseits damit die Problematik der Integration auf EXPRESS-Sprachebene lösbar erscheint, ist die damit offensichtlich verbundene Einbeziehung eines nicht notwendigerweise vollständig relevanten Anwendungsbereiches (Scope) verbunden. Dies erweitert u.U. die Anwendungsbereich des APs und dessen Komplexität des entstehenden Anwendungsprotokolls derart, daß damit wieder das Prinzip eines alles umfassenden Produktmodells entsteht, wie es zu Beginn der STEP-Entwicklung favorisiert worden war. Daß u.U. die Integration der AIMs verschiedener APs formal widersprüchlich sein kann, ist dabei nur von nebensächlicher Bedeutung.

7 Rechnerunterstützung für den elektromechanischen Produktentwurf

Hohe Anforderungen an die Eigenschaften neuer Produkte, die einer Vielzahl von Anforderungsrichtungen entstammen, wie z.B. Funktionalitätserweiterung, Qualität, Reziklierbarkeit und Sicherheit, um nur einige zu nennen, führen zu einer Vielzahl hochentwickelter Entwicklungsmethoden und darauf aufbauend zu einer dementsprechend hohen Zahl von rechnerunterstützten Hilfsmitteln im Bereich der Produktentwicklung.
Das primäre Ziel solcher Entwicklungswerkzeuge, ein spezielles Anwendungsproblem zu lösen, wird dabei oftmals in hohem Maße erfüllt. Weniger zufriedenstellend werden von diesen Werkzeugen jedoch die Anforderungen an die Integration in die EDV-Landschaft eines Unternehmens erfüllt. Die in den vorangegangenen Kapiteln gezeigten Möglichkeiten, aufbauend auf der ISO-10303 stellen einerseits Rahmenbedingungen zur Kopplung bzw. Integration von Anwendungssystemen bereit. Im Rahmen der Arbeit entwickelte oder in jüngster Zeit kommerziell verfügbare Softwaresysteme ermöglichen auf dieser Basis die Lösung der Interfaceproblematik in der technischen EDV-Landschaft.

Damit wird aber zunächst nur die Kopplung bzw. die Integration von bestehenden Anwendungssystemen auf externer Datenebene ermöglicht. Die existierenden Werkzeuge können trotz dieser externen Integration nur die Teilaufgaben der Produktentwicklung lösen, für die sie auch konzipiert wurden.

Neben den in verschiedenen Bereichen zunehmenden Anforderungen an Entwicklungswerkzeuge entstehen jedoch auch völlig neue Anforderungsprofile an Produkte und damit auch an Entwicklungswerkzeuge für solche Produkte. Diese Anforderungsprofile lassen sich unter Umständen nicht mehr durch die Kopplung von insulären Werkzeugen bearbeiten, sondern es wird der Bedarf für völlig neue Entwicklungswerkzeuge sichtbar, die für die gestellten Aufgaben benötigt werden.

Im Folgenden wird aus der Innovation einer Produkttechnologie der Bedarf für solch ein neues Werkzeug abgeleitet. Zunächst wird die neue Produkt- und Produktionstechnologie eingeführt und danach die aktuelle Vorgehensweise im Bereich der Produktentwicklung dieser Produkte skizziert. Aus den identifizierten Defiziten dieser Vorgehensweise kann ein Anforderungsprofil für genau das konzipierte Werkzeug abgeleitet werden. Die Beziehung des Konzeptes des Entwicklungswerkzeuges zur ISO-10303 wird verdeutlicht und die Realisierung des Werkzeuges im Rahmen eines integrierten Ansatzes auf STEP-Basis dargestellt.

7.1 Produktinnovation in der Elektronikproduktion induziert Bedarf nach dreidimensionaler ECAD-Entwurfsfunktionalität

Herkömmliche rechnergestützte Produktentwicklungswerkzeuge stellen zumeist auf Digitalrechnern implementierte Verfahren von Methoden dar, die bereits vor dem Ein-

satz von Rechnern angewandt wurden. Der Fortschritt in der Rechnerunterstützung im Entwicklungsbereich ist primär durch die Implementierung von zunehmend leistungsfähigeren Methoden und Verfahren zu sehen. Totale Innovation in der Rechnerunterstützung der Produktentwicklung ist daher nur bei Existenz völlig neuer Anforderungen an das Entwicklungswerkzeug durch Produktinnovation möglich. Die der Produktinnovation folgende Innovation der Entwicklungswerkzeuge stellt jedoch trotz der Technologiefolge einen wesentliche Einfluß auf die Marktdurchdringung der Produkt-/Produktionsinnovationspartnerschaft dar.

Volkswirtschaftlich maßgebende Industrien sind oftmals durch einen hohen Reifegrad gekennzeichnet. Produktentwicklung und Produktionstechnologie sind hierbei von hoher Effizienz gekennzeichnet. Produkt- und Produktionsinnovationen beziehen sich dabei oftmals nur auf einen Teilbereich des gesamten Produktes bzw. der gesamten Produktionskette. Produktinnovationen, und falls sie zusätzlich Produktionsinnovationen induzieren, stellen im Gegensatz hierzu weitaus höhere Anforderungsfelder an Produkt- und Produktionsentwicklung und die hierfür nötigen auch rechnergestützten Werkzeuge.

Als Beispiel von Produkt- und Produktionsinnovation soll im folgenden die Technologie der räumlichen gespritzten Schaltungsträger (*Molded Interconnect Devices*, MID) herangezogen werden.

7.1.1 MID als Beispiel technologischer Innovation

Die Miniaturisierung elektronischer Komponenten, die durch die Einführung der SMD-Technologie einen Quantensprung erlebt hat, findet jedoch bei Anwendung auf ebenen Schaltungsträgern eine untere Grenze. Diese ist durch die ausschließliche Nutzung speziell bereitgestellter bzw ausgewiesener Bestückflächen für elektronische Bauelemente bestimmt. Diese Bestückflächen finden sich auf dem eigens für diesen Zweck in die Baugruppe aufgenommenen Bauteilträger, der Leiterplatte. Durch konsequente Anwendung der Wertanalyse kann erkannt werden, daß dieses Bauteil neben der in der Baugruppe sekundären Funktion des Bauelementträgers oft keine weitere Funktion besitzt.

Als logische Forderung kann eine wertanalytische Verbesserung des Produkts durch Eliminierung dieses sekundären Bauteils bzw. durch Integration dieses Bauteils mit Bauteilen primärer Funktion erreicht werden. Für einen abgegrenzten Anwendungsbereich kann dies durch eine Integration mechanischer und elektronischer Funktionen in einem dreidimensionalen Schaltungsträger ermöglicht werden.

Die hierbei in den Bauelementträger integrierten mechanischen Funktionen sind zunächst meist den elektronischen Funktionen zugeordnet, wie z.B. Abschirmbleche oder Kühlelemente. Des weiteren werden jedoch auch bereits rein mechanische Funktionen

integriert. Hierunter fallen Befestigungselemente, Lagerelemente, Montagehilfselemente und sogar Gehäusefunktionen mit Designflächen. Die Anwendung frei zugänglicher Flächen als Bauelementgrundflächen bleibt hierbei von einer weitergehenden Nutzung unberührt.

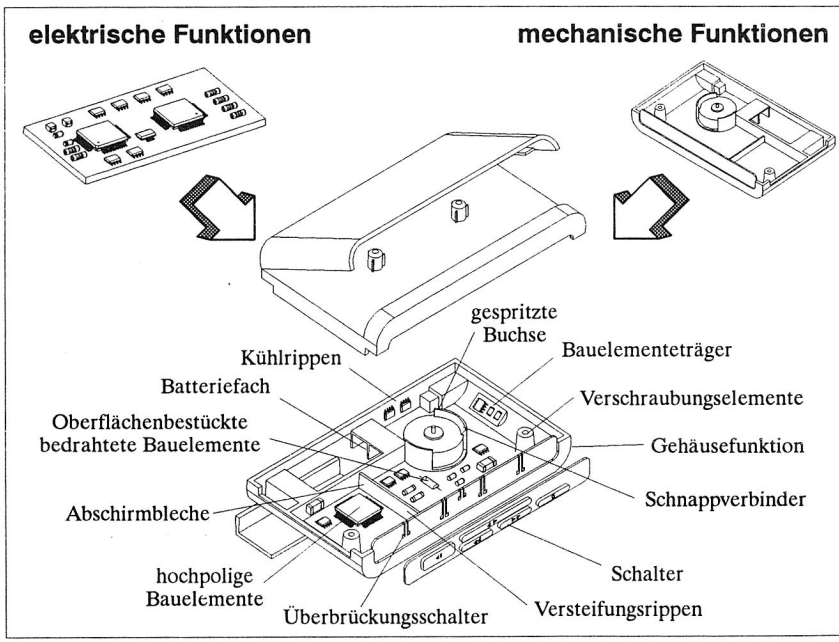


Abb. 47 Tragbarer Cassettenspieler in MID-Technologie (nach /204/)

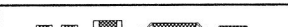
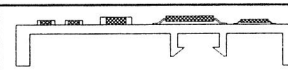
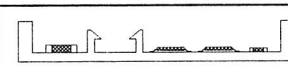
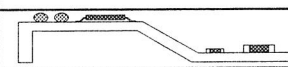
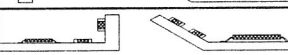
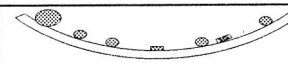
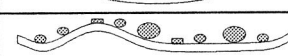
In Abb. 47 ist ein in der MID-Technologie realisierter tragbarer Cassettenspieler dargestellt. Die bezeichneten elektronischen und mechanischen Funktionselemente teilen sich den Gehäusekörper als Bauteil bzw. Bauteilträger.

Der Begriff der Leiterplatte ist dann für einen derart gestalteten Körper nicht mehr adäquat. Die Produktionstechnologie des Bauelementträgers, der im Gegensatz zur konventionellen FR4-Leiterplatte meist im Spritzgußverfahren hergestellt wird, hat eine Begriffswelt wie Molded Circuit Interconnect (MCI), Molded Printed Circuit Board (MPCB), spritzgegossene Leiterplatte mit integrierten Leiterbahnen (SIL) etc., hervorgebracht. Im folgenden soll die vom IPC Molded Board Subcommittee (IPC-MB-380) empfohlene Definition Molded Interconnection Device (MID) verwendet werden /205/:

Molded Interconnection Devices sind spritzgegossene Schaltungsträger aus Thermoplasten mit aufgebrachtem oder integriertem Leiterbild.

Wesentliche Technologien für das Aufbringen des Leiterbildes sind z.B. Laserstrukturierung für Subtraktivverfahren bzw. two-(oder multi-)shot-Moulding als Additivverfahren.

Daß bei Nutzung der dritten Dimension unterschiedliche Qualitäten der Bauelementträger hinsichtlich der Anordnung der Bauelemente auftreten zeigt Tab. 8. Gerade die Prozeßschritte Lotauftrag, Bestückung und Löten sind in ihrer Ausgestaltung wesentlich von der Realisierung des Bauelementträgers abhängig. Werden bei mehreren planparallelen Bestückflächen zunächst nur erhöhte Anforderungen an Bestücksysteme bezüglich der Positionierbarkeit in der Bestücktiefe gestellt, so ist die Bestückung und das Löten an planaren, abgewinkelten Bestückflächen (n x 2D) technologisches Neuland. Weitergehende Miniaturisierung wird in absehbarer Zukunft auch die Nutzung beliebiger Freiformflächen als Bauelementgrundflächen verlangen. Dies wird auch durch die Forderung nach 'function follows form' begründet.

Dimension	Merkmale	Skizze des Aufbaus	Anwendungen
2D	planare Prozeßfläche		konventionelle Leiterplatte
2 ½D	planare Prozeßfläche, 3D-Elemente auf der Rückseite		einfache Gehäuse
	planare Prozeßfläche, 3D-Elemente auf der Prozeßseite		leichte Montage, Modulbauweise,
	mehrere planparallele Prozeßflächen		Fixierung schwerer Bauelemente
n x 2D	mehrere planare Prozeßflächen im Winkel		einfache Gehäuse, platzsparende LP
3D	Regelflächen, z.B. Zylinderflächen		Telekommunikation, KFZ-Technik
	Freiformflächen		Kameras, medizinische Technik

Tab. 8 Klassifikation von Bauelementeträgern nach Dimensionalität der Prozeßflächen (nach /24/)

Das Verfahren des Kunststoffspritzgusses erlaubt vergleichsweise freie Formgestaltung und ermöglicht durch das Zusammenfassen verschiedener Funktionen in dem dreidimensionalen Schaltungsträger die Verringerung der Teilezahl. Damit erfolgt eine Reduzierung der Montageschritte. Die Herstellungskette wird insgesamt verkürzt, und dadurch deren Komplexität verringert. Bei geeigneten Stückzahlen können dadurch die Fertigungskosten gesenkt werden. Die qualitativ und technologisch anspruchsvolle Fertigung elektronischer Komponenten kann erleichtert werden.

Ein weiterer Vorteil der genannten Produkttechnologie betrifft die im Elektronikbereich von gesellschaftlicher und auch von staatlicher Seite erhobene Forderung nach Recyclingfähigkeit von Produkten. Der Einsatz moderner hitzebeständiger thermoplastischer Werkstoffe (PES, PPS, PEI, LCP etc.) gestattet die Verwendung konventioneller Verbindungstechnologien und erfüllt die Forderung nach Umweltverträglichkeit durch

Wiederverwertbarkeit. Darüberhinaus kann Flammwidrigkeit ohne chemische Zusätze erreicht werden.

Die Herstellung der Leiterplatte und das Aufbringen des Leiterbildes sind heute, auch für räumliche Anwendungen, im Labor für kleine Stückzahlen prinzipiell beherrschbar. Die noch zu lösenden Probleme liegen dagegen vor allem in fehlender Rechnerunterstützung im Produktgestaltungsprozeß sowie in der Montage der Molded Interconnection Devices. Die konventionelle Prozeßkette: Lotpaste aufbringen, Bestücken, Reflowlöten ist im allgemeinen dreidimensionalen Fall mit herkömmlichen Automaten nicht mehr realisierbar.

Die Verbindung von mechanischen und elektronischen Aspekten in MIDs läßt diese Produkttechnologie als ein Beispiel mechatronischer Systeme erscheinen. Obwohl der Begriff "Mechatronik" weithin benutzt wird, und sich eine eindeutige Definition bislang nicht durchgesetzt hat, wird im folgenden Mechatronik verstanden als /24/:

Mechatronik beschreibt Produkte bzw. Verfahren zur Entwicklung, Konstruktion und Fertigung von Produkten, die sowohl elektronischen als auch mechanischen Funktionsanteil besitzen, bei denen eine direkte funktionsrelevante Abhängigkeit elektronischer funktionaler Eigenschaften mit mechanischen funktionalen Eigenschaften besteht.

7.1.2 Stand der Technik zur Entwicklung mechatronischer Produkte

Elektronische Produkte, bei deren Entwicklung ein Interesse besteht, sowohl eine logisch-elektrische als auch eine dreidimensionale geometrisch-mechanische Beschreibung bereitzustellen und zu untersuchen, werden heute in stark oder lose gekoppelten EDA und CAD-Systemen entwickelt.

Viele renommierte CAD-Anbieter, vor allem solche mit Produkten im ECAD und MCAD-Bereich, bieten solche Kopplungen auf unterschiedlichem Niveau an. Weniger ausgereifte Lösungen bieten oftmals nur unidirektionale Schnittstellen vom EDA zum MCAD-System an. Hierbei soll im wesentlichen nur eine dreidimensionale Kollisionsüberprüfung von Bauteilen ermöglicht werden. Darüberhinaus werden mit Hilfe dieses Übergangs Untersuchungen in Systemen ermöglicht, die auf rein geometrisch-mechanischer Beschreibung basieren. Hierzu zählt vor allem die Untersuchung thermischer Belastung in FEM-Systemen, wie sie weithin verbreitet ist. Leistungsfähigere Schnittstellen sind darüberhinaus in der Lage auch Positionsveränderungen von Bauelementen im MCAD zurück in das Layoutsystem des EDA-Werkzeugs zu übernehmen.

Innerhalb des Elektronikdesigns wird die gewünschte elektronische Funktion durch Komposition elementarer Funktionen synthetisiert. Dies erfolgt zweistufig in getrennten Modulen innerhalb der EDA-Systeme: In logischen Entwurfssystemen werden elektri-

sche Elementarfunktionen, durch grafische Symbole repräsentiert, in einer Arbeitsfläche rein logisch angeordnet, und miteinander durch Linien verbunden, was der elektrischen Verschaltung entspricht. Dieser logische Entwurf wird dann oft in einem vom Logik-Entwurfswerkzeug getrennten Layoutsystem, weiterverarbeitet. Das Layoutsystem liest eine sogenannte Netzliste ein, die die logischen Grundelemente sowie deren elektrische Verschaltung darstellt. In dem Layoutsystem wird nun das Packaging, das Zuordnen logischer Elementarfunktionen zu realen physikalischen Bauelementen vorgenommen. Dann werden die Bauelemente auf dem Bauelementträger positioniert und die jeweiligen Verbindungen geroutet. Dabei können die umfangreichen Möglichkeiten moderner Layoutsysteme wie Autorouter, Netlist Connectivity Checks oder Design Rule Checks eingesetzt werden (Abb. 48).

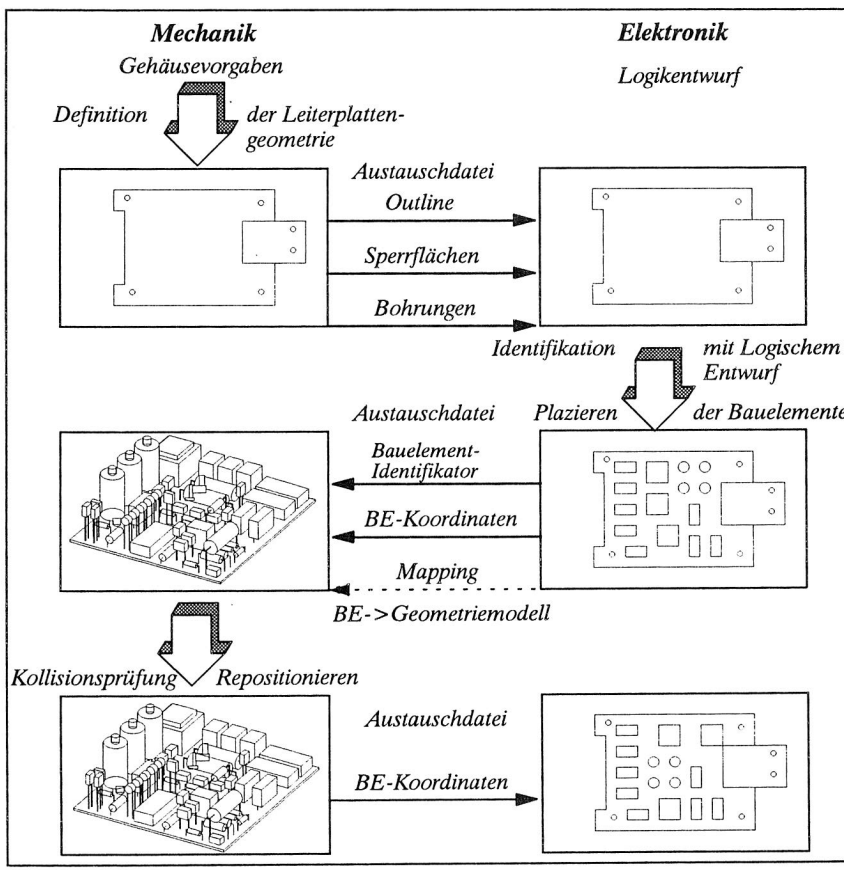


Abb. 48 Vorgehen beim kooperativen Entwurf unter Nutzung der Kopplung zwischen ECAD- und MCAD-Systemen

Innerhalb der mechanischen Gerätekonstruktion wird das Gehäusedesign bereitgestellt. Ist zu erwarten, daß die geforderte Form des Gehäuses Speziallösungen des Bauelementträgers nötig macht, werden aus der Gestaltung des Produktgehäuses die geometrischen Abmessungen der Leiterplatte abgeleitet oder bei räumlichen Schaltungsträgern (flexible, starrflexible, Bendflex, 3D-MIDs etc.) durch Abwicklung der dreidimensionalen Geometrie die Aufgabe auf die ebene Problemstellung reduziert. Ebenfalls bereits im MCAD können Bereiche definiert werden, in denen die Plazierung von elektronischen Bauteilen vermieden werden muß oder wo gehäusekritische Komponenten positioniert werden sollen. Die Befestigung und elektrische Verbindung der Leiterplatte zu ihrem Träger durch Bohrungen, Steckerleisten, Kabelstränge, flexible Leiterbahnen etc. kann ebenfalls bereits in der mechanischen Konstruktion festgelegt werden.

Die für die Elektronikkonstruktion relevanten Daten wie Leiterplattengeometrie, Bereichsdefinitionen, Befestigungsformelemente, spezielle Bauteilpositionen etc. werden rechnerintern vom MCAD- an das ECAD-System übergeben. Dabei werden meist systemeigene anwendungsorientierte Beschreibungsformate zur Übertragung genutzt. Die elektronischen Bauteile werden dann im EDA unter Berücksichtigung elektrischer Randbedingungen manuell oder automatisiert plaziert.

Die ermittelten Bauteilpositionen der elektrischen Bauelemente werden nun ins mechanische System übertragen. Dort wird durch anwendungsspezifische Kopplungsmodule automatisiert eine dreidimensionale Repräsentation der Leiterplatte erzeugt. Mit dieser Repräsentation kann nun bei komplexen Gehäusegeometrien Kollisionsfreiheit und geforderte Mindestabstände sichergestellt werden. Sollten Änderungen des Layouts nötig sein, ist dies im EDA-Layoutsystem durch Rückspielen der Positionsdaten oder manuelle Korrektur zu beheben. Danach kann eine erneute Datenübergabe ins MCAD nötig sein.

Die automatisierte Synthese einer dreidimensionalen Repräsentation basiert im Wesentlichen auf einem Rückgriff auf Bibliotheken dreidimensionaler Bauelemente und deren Anordnung, die dem zweidimensionalen Layout entnommen werden kann. Diese Bauelemente sind für elektrische Anwendungen im zweidimensionalen in allen EDA-Systemen üblich.

Falls durch Analysen in den spezifischen Analysewerkzeugen Schwachstellen im Leiterplattenlayout erkannt werden sollten, muß eine oder weitere Regelschleifen zurück ins ECAD durchgeführt werden, bis Funktion und wirtschaftliche Fertigbarkeit sichergestellt sind. Aus den jeweils am besten geeigneten Produktmodellen werden die Fertigungsunterlagen generiert. So bietet sich z. B. die Ausgabe der Photoplotdaten aus dem ECAD und die Fertigungs- oder Montageprogramme aus dem MCAD an. Darüberhinaus sind auch Testdaten und Dokumentationen bereitzustellen.

7.1.3 Komplexitätsanalyse von MID-Produkten

Die Realisierung eines Prototyps eines Entwurfssystems für dreidimensionale spritzgegossene Schaltungsträger ist im wesentlichen abhängig von der Fähigkeit bereits existierende Softwaremodule zu einem Gesamtsystem integrieren zu können. Die Entwicklungsaufwände für Teilmodule wie Geometrieverarbeitung und Graphische Darstellung lassen sich z.B. von deren von rein mechanischen Anwendungssystemen abschätzen, die Umfänge von mehreren hundert Mannjahren betragen.

Die Analyse bekannter MID-Produkte zeigt (siehe Abb. 49), daß die bisherigen Entwicklungen vorm allem durch geometrisch/mechanische Anforderungen geprägt sind. Die elektronische Funktionalität wird dabei durch wenige aber hochintegrierte elektronische Bauelemente realisiert. Der Hauptschwerpunkt der Anforderung an die Entwicklung von MID-Produkten liegt damit offensichtlich in der Gestaltung des spritzgegossenen Schaltungsträgers. Die elektrische Funktionalität wird durch einen einfachen Schaltungsaufbau weniger Bauelemente realisiert.

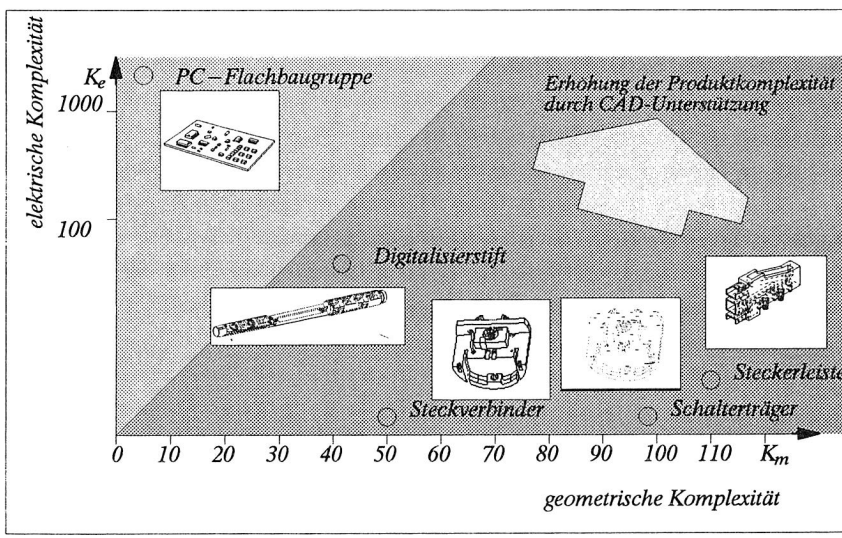


Abb. 49 Verhältnis der geometrischen zur elektrischen Komplexität aktueller MID-Produkte

Dieses Komplexitätsverhältnis kann durch das Verhältnis der gewichteten Anzahl der elektronischen Bauelemente sowie der Anzahl der Verbindungen mit der Anzahl der als Bestück- und Leiterbahnflächen benutzten Produktflächen bestimmt werden. Es ergibt sich:

Komplexitätsfaktor $Q = (V(a \times BE_i) + V(b \times Netz_i)) / (c \times Fläche_i)$

Die Gewichtungsfaktoren a, b und c sind heuristisch aufgrund von Ähnlichkeitsfaktoren gewählt. Der Wert a für Bauelemente beträgt z.B. für ein kubischen Widerstand willkürlich 1 und für ein SOT-8 Bauelement z.B. 1,8. Für eine elektrische Verbindung ohne Verzweigung mit N geraden Abschnitten wird der Wert N * 1 berechnet.
Der Gewichtungsfaktor c der Nutzflächen wird für eine planare Fläche willkürlich auf 1 gesetzt sowie für zylindrische Flächen auf 1,2, für konische auf 1,5, sphärischen auf 1,8 und Freiformflächen auf 2,5 gesetzt.

In Abb. 49 wurden die Komplexitätsfaktoren für einige MID-Produkte aufgetragen. Es ist zu erkennen, daß der Komplexitätsverhältnis Q der MID-Produkte, aufgetragen auf einer Skala getrennt für den elektrischen und den mechanisch/geometrischen Komplexitätsfaktor die Annahme stützt, MID-Produkte sind im wesentlichen geometrisch determiniert. Als Gegensatz dazu wurde nach den identischen Bewertungsverfahren eine hochkomplexe, planare Leiterplatte der Computertechnologie analysiert. Diese Leiterplatte trägt sich oben links in dem Tableau ein.

Ein Konzept eines Entwicklungssystems für MID-Produkte muß diesem Umstand Rechnung tragen und die geometrische Determiniertheit der MID-Produkte wesentlich unterstützen. Als Basis eines Entwurfswerkzeuges bietet sich daher ein geometrieverarbeitendes System an, zu dessen Funktionalität die elektrischen Entwurfsfunktionalitäten, die über die reine Geometrieverarbeitung hinausgehen, hinzugefügt werden.

7.1.4 Analyse der Defizite existierender Konzepte - Pflichtenheft für ein Neukonzept

Die in Kapitel 7.1.2 vorgestellten Kopplungen von Mechanik- und Elektronikentwurfssystemen ermöglichen die Anwendung von spezifischem Anwendungsfeatures der jeweils genutzten Systeme. Funktionale Erweiterungen für den elektromechanischen Entwurf in den beteiligten Systemen existieren zumeist nicht oder sind auf einfache, in den beteiligten Systemen darstellbaren Funktionen beschränkt.

Die aus den MID-Produkten analysierbaren Anforderungen an ein Entwicklungswerkzeug können nicht aus der Kopplung von bestehenden Werkzeugen hervorgehen. Der Bedarf für ein neues Werkzeug mit dedizierten Entwicklungsfunktionen wird nötig.
Die Anforderungen an ein solches Werkzeug, die ein erfolgreiches Konzept von gekoppelten Systemen verhindern, sollen im folgenden kurz dargestellt werden. Für jeweils beide Partnersysteme im Sinne einer Kopplung können Restriktionen angegeben werden, die durch Kopplungen nicht erfüllt werden können:

- Die Gestalteigenschaft von MID sowie deren Produktionstechnologie die oftmals durch Spritzguß thermoplastischer Werkstoffe realisiert wird verlangen nach dreidimensionaler Repräsentation des Produktes. Dies kann von konventionellen ECAD-Systemen nicht erreicht werden. Würde nur das Mechaniksystem eine dreidimensionale Darstellung unterstützen, so müßte beim Übergang zum Elektroniksystem

eine u.U. auch partielle Abwicklung der bestückten Flächen vorgenommen werden. Nicht total abwickelbare Flächen sowie multi-shot Schaltungsträger ließen sich damit überhaupt nicht realisieren.

- Das logisch/elektronische Teilmodell einer MID-Komponente kann nicht in mechanischen Anwendungssystemen dargestellt werden. Die Abstraktion von Logischer/ Elektrischer Information ist für viele MID-Anwendungen jedoch nicht zulässig. Die Generierung von Fertigungsinformation für Laserstrukturierung basiert jedoch auf der Existenz der Leiterbahnstrukturinformation im dreidimensionalen mechanischen Modell. Ebenfalls sind Leiterbahninformationen für die Generierung von NC-Datensätzen für die Bereitstellung von Spritzgußwerkzeugen nötig.

7.2 Anwendungsprotokoll 210 als Grundlage für ein integriertes dreidimensionales Entwurfssystem

Bisher wurde innerhalb dieses Kapitels der Bedarf für ein neues Entwicklungswerkzeug hergeleitet. In früheren Kapiteln wurde der Einfluß der Norm ISO10303-STEP auf die Entwicklung technischer EDV-Systeme dargestellt. Im folgenden soll nun der Brückenschlag zwischen einerseits der ISO10303 (bzw. eines Teils davon) und der durch MID induzierten Anforderungen an Entwicklungswerkzeuge vorgenommen werden.
Die in den STEP-Anwendungsprotokollen festgeschriebenen Anwendungsbereiche ermöglichen es, technische Anwendungssysteme zu realisieren, die einerseits durch den in der Norm definierten Gültigkeitsbereich beschrieben werden, andererseits aber durch den standardisierten Produktmodellaspekt leicht einer Integration in Produktdatenbanken zugänglich sind.

Als relevantes STEP-Anwendungsprotokoll für die Entwicklung von Schaltungsträgern ist das AP210 /62/ gegeben, das im folgenden die produktmodellorientierte Basis der Entwicklungen sein soll.

7.2.1 Anwendungsbereich des Anwendungsprotokoll 210

Obwohl die in den STEP-Anwendungsprotokollen zunächst bearbeiteten Anwendungsbereiche primär den Problembereichen der Konstruktion im allgemeinen mechanischen oder der spezialisierten Teilefertigung des Maschinen- und Fahrzeugbau orientiert waren, wurden auch die Austauschproblematiken des Elektronikindustrie durch die Entwicklung eigener, auf die Elektronikentwicklung und Fertigung hin orientierte Anwendungsprotokolle, initiiert. Diese Problematik wird zur Zeit in vier Anwendungsprotokollen bearbeitet:

- das Anwendungsprotokoll 210 zur Beschreibung von Leiterplatten /62/.

- das Anwendungsprotokoll 211 zur Beschreibung des Tests, Diagnose und Nacharbeit von Elektronischen Komponenten .
- das Anwendungsprotokoll 212 zur Beschreibung von Elektrischen Anlagen /63/.
- das Anwendungsprotokoll 220 zur Beschreibung der Fertigung von Leiterplatten.

Während sich das AP212 primär auf die Beschreibung von elektrischen Anlagen bezieht, und die anderen genannten APs eher der Fertigung elektronischer Komponenten zuzuordnen sind, stellt das AP210 genau den Ausschnitt eines Produktmodells dar, der die Konstruktion von Leiterplatten unterstützt. Das Anwendungsreferenzmodell der AP210 enthält acht Funktionalbereiche (*units of functionality*) die in Abb. 50 dargestellt sind.

Obwohl das AP210 die zur Zeit industriell relevante Produktionstechnologien, bzw. damit herstellbare Produkttechnologien wie THT (Through Hole Technology, bedrahtete Bauelemente), SMT (Surface Mount Technology, Technologie der oberflächenmontierten Bauelemente), gehäuste und ungehäuste Bauelemente, gedruckte Schaltungen usw. unterstützt, bleiben neueste Entwicklungen bisher unberücksichtigt.

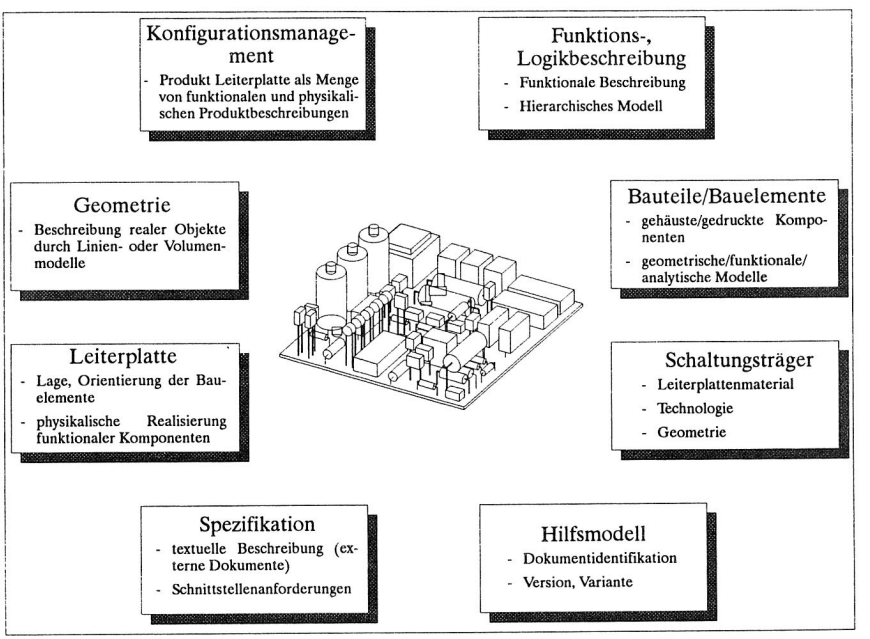


Abb. 50 Bestandteile der Anwendungsreferenzmodelles des Anwendungsprotokolls 210 der ISO10303

Das vorliegende Dokument des AP210 /62/ nimmt noch keine Rücksicht auf die durch die oben eingeführte MID-Technologie und deren Anforderungen. In Kapitel 7.2.3 wird

das Anforderungsreferenzmodell (ARM) des AP210 um die in Kapitel 7.1.4 als Defizite identifizierten Konzepte erweitert.

7.2.2 Einfluß der Bauelementbibliotheken - Normbauelementbibliotheken

Der Entwurf von elektronischen Komponenten ist im Vergleich zur Konstruktion mechanischer Produkte deutlich stärker durch die Komposition verfügbarer Komponenten gekennzeichnet. Die Verwendung elektronischer Bauelemente in elektronischen Produkten, speziell im Bereich der Leiterplattentechnologie stellt die Entwicklungsmethodik als solches dar. Die Entwicklungsproblematik ist damit vielfach auf eine Konfigurationsproblematik reduziert. Die Bereitstellung der Komponenten erfolgt für die Fertigung der Produkte durch eine Kunden-Lieferantenbeziehung, für den Bereich der Produktentwicklung stellen die Anbieter von ECAD-Systemen umfangreiche und zumeist kundenspezifisch erweiterbare Bauelementebibliotheken bereit.

Während der Entwicklung der ISO-10303 wurde, wenn auch zunächst für den Bereich der mechanischen Konstruktion, die Problematik von Normteilbibliotheken erkannt und eine Arbeitsgruppe zur Definition eines Standards in diesem Bereich gebildet. Die Arbeitsgruppe 2 (WG2) des TC 184 erarbeitet innerhalb der ISO-13584 eine Norm zur neutralen Beschreibung von Normteilebibliotheken (Parts Libraries, P-Lib). Obwohl die Definition der Normteilebibliotheken nicht innerhalb der ISO-10303 stattfindet, besteht eine enge Beziehung zwischen beiden Normen (bzw. Normvorschlägen), die sich u.a. in der Modellierung der Normteilebeschreibung mit Hilfe der ISO-10303 Modellierungssprache EXPRESS dokumentiert.

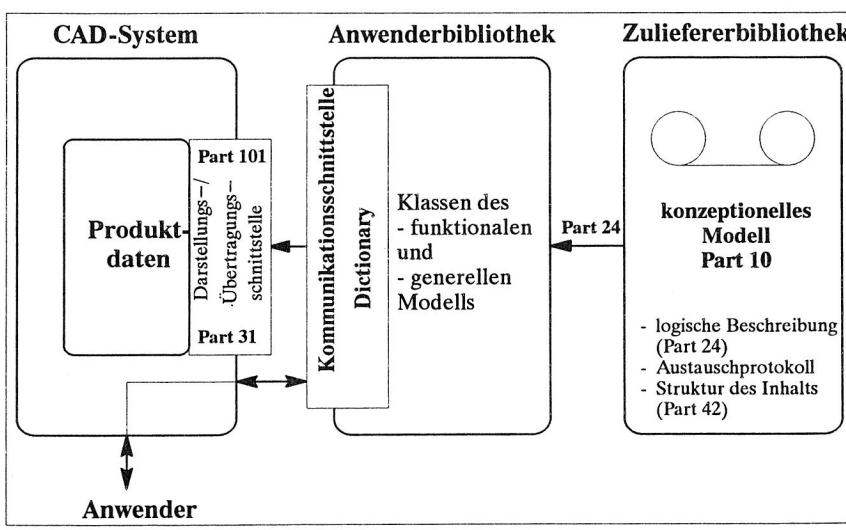


Abb. 51 Konzept zur Nutzung von Bauelementbibliotheken der ISO 13584 (nach /184/)

Der Normvorschlag zur ISO 13584 besteht aus einer Reihe von einzelnen Teilen, von denen der Teil ISO 13584-10 zur Beschreibung allgemeiner Konzepte, der Teil ISO 13584-42 zur Beschreibung zu Struktur und Inhalt des Verzeichnisses (Dictionary) der Teilebibliothek dient. Darüberhinaus wird eine Kommunikationsschnittstelle zum Zugriff auf den Bibliotheksindex definiert, der einen einheitlichen Zugriff auf den Inhalt der Teilebibliothek ermöglicht, wie er zum qualifizierenden Zugriff während des Konstruktionsvorganges genutzt wird (siehe Abb. 51).

Die Realisierung der Bereitstellung von Wiederholteilebibliotheken im Format der STEP-Austauschdateien impliziert die Problematik der dateiübergreifenden Referenzierung. Innerhalb einer STEP-Austauschdatei sind die Instanzbezeichner eindeutig. Über den Kontext der Datei hinaus ist die eindeutige Referenzierung nicht möglich. Für Teilebibliotheken bedeutet dies, daß Bibliotheken nicht modular bereitgestellt werden können, wenn derartige eindeutige Referenzen über Dateigrenzen nicht garantiert werden können.
Innerhalb der ISO 13584 wurde daher das Konzept einer eindeutigen Identifikation eines Elementes einer Teilebibliothek bereitgestellt: Jedes Element eines Bibliotheksindexes erhält einen weltweit eindeutigen Code, mit dem dieses Element referenziert werden kann. Als Einstiegscodes solcher Referenzen werden Identifikationscodes für die Anbieter der Elemente über die ISO vergeben.

Ein wesentliche Anforderung der Entwicklung des AP210 besteht in der Notwendigkeit, Leiterplattenmodelle von der expliziten Darstellung aller in einem Leiterplattenmodell inhärenten elektrischen Bauelemente zu trennen. Hochvolumige Datentransfers von "allgemein" zugänglichen Bauelementinformationen sind so zu vermeiden.
Darüberhinaus kann über diesen Referenzmechanismus die Problematik der Verfügbarkeit von Bauelementen gelöst werden: Ein nicht mehr verfügbares Bauelement wird durch eine nicht auflösbare Referenz deutlich. Über die gegebenen Anforderungen an das Bauelement kann aus der Bibliothek nach einem gleichwertigen Alternative gesucht werden.

7.2.3 Erweiterungen des AP210 um MID-spezifische Anforderungen

Mit Hilfe von MID-Komponenten werden innerhalb eines Produktes solche Anteile realisiert, die einen elektrischen-logischen und solche, die einen mechanisch-geometrischen Funktionsanteil besitzen. Während die Beschreibung geometrischer Sachverhalte prinzipiell wenig Schwierigkeiten macht, wie dies in geometrieverarbeitenden Systemen und in den STEP-Geometriemodellen gezeigt wird, andererseits die elektrisch-logische Modellbildung ebenfalls weithin akzeptiert wird, ist durch die reine Erweiterung eines Leiterplattenmodells auf dreidimensionale Darstellung, wie dies in AP210 vollzogen wird, nicht ausreichend, um MID-Komponenten zu beschreiben.
Die in elektrische-logische und geometrisch-mechanische Modellbildung allein wird

von MID-Komponenten nicht berührt, allein die Beziehungen zwischen den Repräsentationen elektrischer und geometrischer Funktionsträger wird hiervon berührt.

Anhand der identifizierten Defizite des Anwendungsreferenzmodells des AP210 wurden für die relevanten Teilbereiche Erweiterungen vorgeschlagen. Diese sind den Funktionalbereichen (Uofs) Geometrie, Bauteile, Leiterplatte und Schaltungsträger (siehe Kapitel 7.2.1) zuzuordnen. Im einzelnen wurde in diesen Funktionalbereichen die folgenden Erweiterungen vorgeschlagen, die aus den MID-spezifischen Anforderungen an die Produkttechnologie erwachsen:

- Funktionalbereich Geometrie:
  Der Funktionalbereich wurde für die Anwendbarkeit hinsichtlich der Gestaltausprägung von Leiterplatten erweitert. Explizit bedeutet dies, daß die Einschränkung auf ebene Schaltungsträger aufgehoben wurde. Für das Mapping auf das Anwendungsinterpretierte Modell (AIM) kann damit ein Mapping auf beliebige Ausprägungen des AIM-Elementes *Shape_Representation* erwartet werden.
- Funktionalbereich Bauteile:
  Für den Funktionalbereich der Bauteilbeschreibung wurde die Verwendung von Referenzen aus Bauteilbibliotheken nach ISO 13584 dringend angeraten. Da dieser Teilbereich einem LP-Systementwurf einerseits konstruktiv selten zugänglich, andererseits dessen Verwendung innerhalb der LP-Konstruktion elementar ist, wurde die Zuordnung dieses Aufgabenbereiches an die Bereitstellung eines Bauteilekataloges bzw. der Referenzierung aus diesem Katalog vorgenommen.
- Funktionalbereich Leiterplatte:
  Der Funktionalbereich 'Leiterplatte' (*Printed Circuit Assembly*, PCA) des ARM des AP210 beschreibt, anders als zunächst aus dem Titel zu vermuten, die Ausprägung einer gedruckten und bestückten Leiterplatte. Die Positionsinformation der Bauelemente auf einer dreidimensionalen Leiterplatte im AP210 genügt den Anforderungen der MID-Technologie. Die Lage und Orientierung von Bauelementen ist durch die Festlegung von 6 Freiheitsgraden darstellbar.
- Funktionalbereich Schaltungsträger:
  Der Funktionalbereich Schaltungsträger (*Printed Circuit Board*, PCB) definiert das unbestückte Board (*bare board*) mit aufgebrachten Leiterbahnen, leitenden Flächen, Durchkontaktierungen (Vias) usw.. Während die Anforderungen des ARM für ebene Leiterplatten einen Großteil der Anforderungen für MID-Schaltungsträger erfüllt, stellen die Möglichkeiten der MID-Technologie in diesem Bereich höhere Anforderungen:
  Leiterbahnen für MID-Schaltungsträger müssen im Gegensatz zu konventionellen Leiterplatten nicht nur auf Leiterbahnober- und -unterseite darstellbar sein, sondern auch ein Beschreibung im Innern eines Schaltungsträgers zulassen. Die Bahn eines Leiterbahnzuges kann dabei eine beliebige abschnittsweise verbundene Menge von Linienzügen sein. Der Querschnitt von Leiterbahnzügen ist darüberhinaus an wenig Einschränkungen gebunden. Zum einen kann eine abschnittsweise kon-

stante, linear veränderliche, im allgemeinen aber durch einen beliebig funktional beschreibbaren Querschnitt darstellbar sein.
Darauf aufbauend muß an die Geometriebeschreibung von Leiterbahnen die geometrische Forderung gestellt werden, daß Kontakt oder Durchschnitt von Leiterbahnzügen, die unterschiedlichen Netzen bzw. Potentialen angehören, verboten sind. Mit dieser Forderung wird Bezug zu der logisch/elektrischen Sicht auf das Leiterplattenmodell genommen.
Weiterhin muß die Leiterbahngeometrie jedoch dahingehend eingeschränkt werden, daß der Leiterbahnkörper zumindest flächigen Kontakt zu dem tragenden Basiskörper haben muß. Für interne Leiterbahnverläufe gilt der flächige Kontakt für alle Flächen des Leiterbahnkörpers.

Mit den hier identifizierten Defiziten sowie den vorgeschlagenen Anforderungserweiterungen müssen während des ARM-AIM Mapping-Vorganges (siehe Kapitel 6.1.2) der AP210-Entwicklung die hinreichend gestalteten Elemente der Integrated Resources ausgewählt werden. Für die Belange der Geometriebeschreibung sind hierfür in den Geometriemodellen geeignete Elemente vorhanden. Die modulare Unterstützung logischer Verbindungen wird durch das in im folgenden Kapitel definierte Partialmodell 'Konnektivität' ermöglicht.

7.2.4 Definition eines AICs 'Konnektivität'

Die in den Teilen der Generic Resources und AICs vorhandenen Elementen zur Beschreibung von technischen Sachverhalten stützen sich vornehmlich auf die Darstellung organisatorischer, geometrischer oder mathematischer Bereiche.

Die Integration logisch-elektrischer Sachverhalte in die STEP-Partialmodelle verlief nicht derart erfolgreich, wie dies oftmals vermutet wurde /83/. Gründe hierfür sind einerseits in mangelnder Budgetierung zu suchen, andererseits aber sicher auch durch die weniger starke Nachfrage solcher Partialmodelle begründet. Darüberhinaus stellt die in der STEP-Entwicklungsmethodik begründete Modularisierungsproblematik eine weitere Hürde dar. Im Part 103 der Application Resources /59/ wurde begonnen, ein Partialmodell für hierarchische Konnektivität zu definieren. Die Entwicklungen für das AP210 /62/ und das AP212 /63/ wären geeignete Kandidaten zur Nutzung dieses Partialmodells gewesen. Ende 1995 wurde die Entwicklung jedoch eingestellt. Der Normvorschlag befand sich zu diesem Zeitpunkt in einem unbrauchbaren Zustand, stellt jedoch einige Anforderungen an das Partialmodell dar.

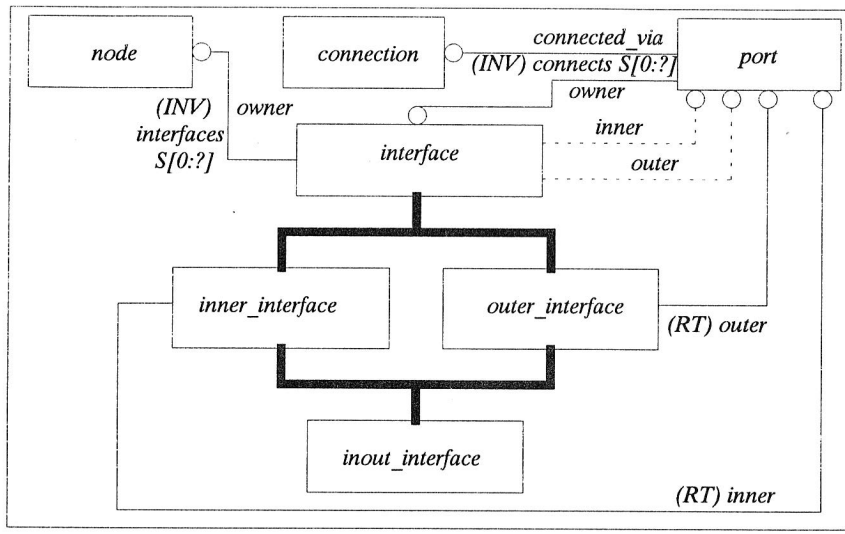


Abb. 52 EXPRESS-G Diagramm des Partialmodells hierarchical_connectivity_schema

Um die Interoperabilität zwischen den APs 210 und 212 im Bereich des Logikmodells zu unterstützen, wurde auf Basis des Part 103 /59/ ein vereinfachtes Partialmodell für logische Konnektivität entwickelt. Im Gegensatz zu /59/ wurde dabei auf jegliche Informationsanteile verzichtet, die den (Informations-, Material- oder Energie-) Fluß über die durch die Konnektivität verbundenen Elemente beschreibt. Damit wird eine weitergehende Modularisierung erreicht. Anwendungsbereiche des Partialmodells können nun ebenfalls Modelle sein, die Material oder Energie über die verbundenen Knoten transportieren.

Als relevante Elemente, die zur Beschreibung einer Konnektivität nötig sind, werden folgende Objekte identifiziert:
Der Knoten (*node*), der die Objekte beschreibt, die mit Hilfe der Konnektivität verbunden werden sollen. Der Knoten besitzt eine Menge von Interfaces (*interface*). Ein Interface selbst besitzt optional innere und äußere Ports. Ein Port (*port*) wiederum stellt einen Verbindungspunkt zwischen einer Verbindung (*connection*) und einem Interface dar. Da ein Interface entweder innere, äußere, innere und äußere oder aber keine Ports besitzen kann, wurden für die ersten zwei Fälle jeweils Spezialisierungen des *interface* eingeführt, um die optionalen inversen Beziehungen zwischen den vorhandenen Ports und dem jeweiligen Interface besser modellieren zu können. Eine weitere Spezialisierung in ein Interface, das sowohl innere als auch äußere Ports besitzt (*inout_interface*), erzwingt damit eine inverse Beziehung für beide Ports.

Mit dem vorliegenden Modell läßt sich ein Element einer hierarchischen elektrischen Verschaltung wie folgt beschreiben:

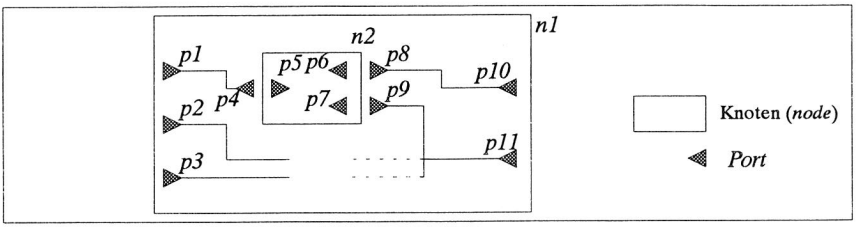


Abb. 53 Blockschaltbild eines hierarchischen Konnektivitätsmodells

Das Blockschaltbild aus Abb. 53 besteht aus zwei Knoten (*node*) n1 und n2. Diese können beliebig gestaltete Funktionseinheiten beschreiben. Der Knoten n1 besitzt 5 Schnittstellen (*interface*), die alle ein ein *inner_interface* dargestellten. Die äußeren Teile der Schnittstellen von n1 seien in diesem Beispiel nicht von Interesse. Die drei Interfaces des Knoten n2 stellen allesamt *inout_interfaces* dar und werden jeweils durch ein assoziiertes Paar von Ports repräsentiert.

Von der Bezeichnung der identifizierten Objekte wurde wissentlich abstrahiert. Den Entitäten zur Beschreibung von Repräsentation aus den Generic Resources (*representation_item*) liegen Attribute bei, die die Bezeichner solcher Repräsentationen liefern können.

7.3 Prototyp eines dreidimensionalen Entwurfssystem

Im folgenden soll das Konzept eines Moduls zur Unterstützung des Entwurfes von MID-Komponenten dargestellt werden. Mit den skizzierten Anforderungen an die Modellbildung von dreidimensionalen Schaltungsträgern sowie den mit STEP möglichen Integrationsspielräumen kann ein in einen betrieblichen Entwicklungsablauf integrierbares CAD-Modul realisiert werden. Die wesentliche Orientierung soll dabei auf der Layoutunterstützung von beliebig geformten Schaltungsträgern liegen.

7.3.1 Konzeptionelle Einbettung des Prototypen in ein STEP-Anwendungssystem

Mit den in Kapitel 5 vorgestellten Geometriemodellierern Parasolid und Acis standen zwei leistungsfähige Geometriemodellierer zur Verfügung, die beide als Basis eines MID-Entwicklungswerkzeuges dienen könnten. Die Entscheidung für einen der beiden Systeme muß aus einer Vielzahl von Kriterien bestimmt werden, die die Funktionalität bzw. Leistung der Entwicklungswerkzeug beeinflussen. Da sich die beiden Systeme in ihrem internen Aufbau und in der Unterstützung unterschiedlicher Geometriekonzepte sehr ähnlich sind, kommen darüberhinaus Kriterien zur Bewertung, die eher von softwa-

retechnologischer Natur sind. Die Entscheidung für den Modellierer Acis wurde durch Berücksichtigung der folgenden Kriterien getroffen:

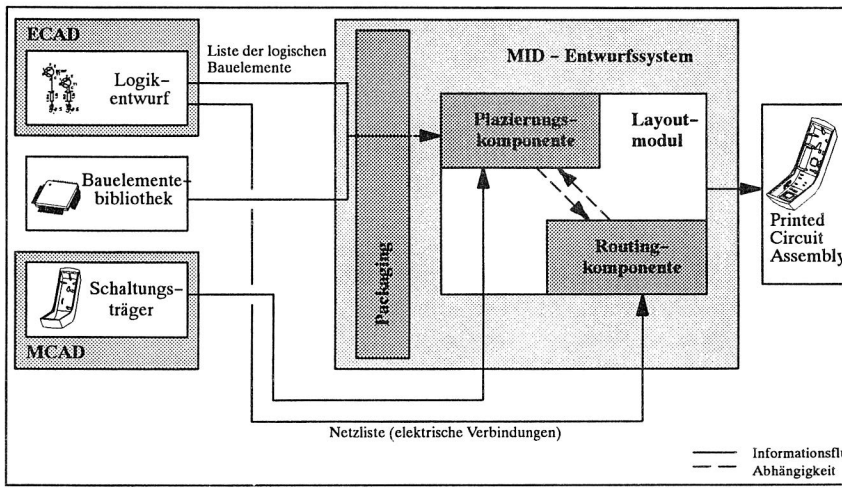


Abb. 54 Konzept eines MID-Entwurfssystems

- Einfachere und flexiblere Programmierschnittstellen durch Klassenschnittstelle und API-Funktionen.
- Höhere Performanz.
- Einfachere Anbindung eines graphischen Darstellungssystems sowie dessen Unterstützung durch ein eigenes Toolkit.
- Bessere Unterstützung benutzerdefinierte Attribute.

Der Modellierer Acis soll also das softwaretechnische Rückgrat eines Entwicklungsmoduls sein, das sich in eine Gesamtarchitektur wie folgt eingliedert:
Neben dem im Geometriemodellierer Acis gehaltenen geometrischen Partialmodell eines Schaltungsträgers existiert ein eigen definiertes elektrisches Partialmodell einer elektronischen Schaltung, das im Wesentlichen aus dem in /62/ definierten elektrischen Modell besteht. Beide Partialmodelle sind über ein SDAI aus einer STEP-Datenbasis zugänglich. Das Acis-Geometriemodell wird von der Anwendungsseite des Modellierers transparent aus der Datenbasis geladen. Der elektrische Partialmodellanteil hingegen ist nur direkt aus der Datenbasis über SDAI-Zugriffe zugänglich. Als STEP-Datenbasis dienen das in Kapitel 3.3.6 vorgestellten Modul. Alternativ ist die native Dateischnittstelle des Modellierers Acis und die in Kapitel 3.3.3 vorgestellte STEP-Austauschdateischnittstelle verfügbar.

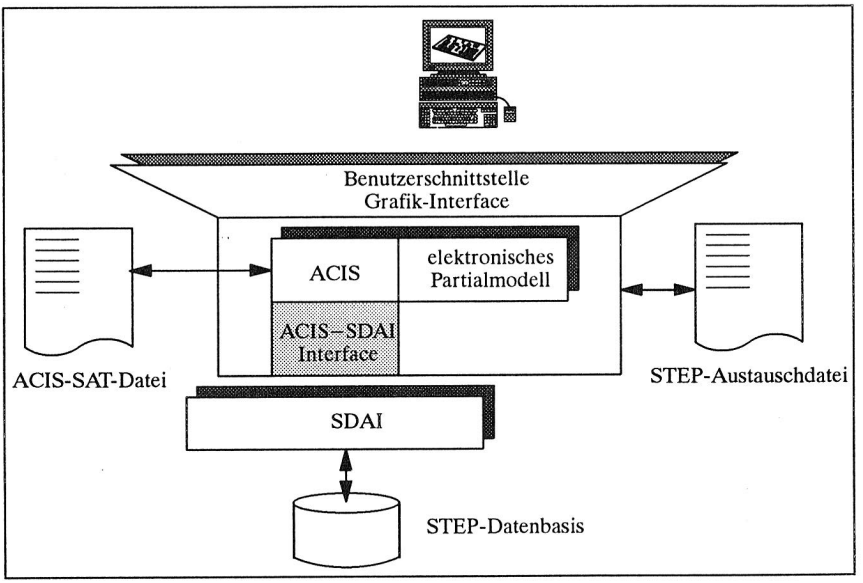


Abb. 55 Gesamtarchitektur des MID-Entwicklungswerkzeuges

Als die beiden wesentlichen funktionalen Anforderungen an ein Entwicklungswerkzeug, die über die bekannten Funktionalitäten existierender zweidimensionaler Entwurfssysteme sind einerseits eine Komponente zur Bauelementeplazierung auf beliebig geformten Schaltungsträgern und eine Komponente zum Routing (Entflechten) von Leiterbahnen auf dreidimensionalen Schaltungsträgern. Module zur Realisierung dieser Funktionalitäten werden in den folgenden beiden Kapiteln vorgestellt.

7.3.2 Bauelementeplazierung auf dreidimensionalen Schaltungsträgern

Als Eingangsinformationen für das Layoutmodul ist einerseits ein Geometriemodell des dreidimensionalen Schaltungsträgers sowie ein aus der Elektronikentwicklung hervorgegangenes Elektronikmodell (siehe Abb. 54). Das Elektronikmodell muß sich jedoch bereits in einem Zustand befinden, in dem die elektrischen Funktionsträger realen Bauelementen zugeordnet sind. Der hierfür nötige Packagingvorgang liegt außerhalb der hier betrachteten Problematik.
Das Geometriemodell kann von außen durch Acis SAT-Dateien[18] oder STEP-Austauschdateien in das System eingebracht werden. Dabei können alle STEP-Austauschdateien, die Partialmodelle enthalten die durch die AICs beschrieben werden, benutzt werden.

18. SAT-Dateien sind die nativen Dateien des Geometriemodellierers Acis

Der primär geladene Körper wird als Schaltungsträger identifiziert. Die für eine Schaltungsrealisierung nötigen Bauelemente werden als Menge von Bauelementen in das System geladen und zur Plazierung herangezogen. Die Plazierung der Bauelemente müssen dabei der Restriktion gehorchen, daß jedes Bauelement mit der für ihm charakteristischen Basisfläche auf einer Fläche des Schaltungsträgers positioniert werden müssen.

Die Basisfläche eines Bauelementes ist entweder bekannt, falls das Bauelement aus der Bibliothek entnommen wurde, oder sie kann vor Beginn des Plazierungsvorganges interaktiv bestimmt werden. das System schlägt in letzerem Falle eine Ebene vor, die die Ebene beschreibt, die normal zur lokalen XY-Ebene des Bauelementes liegt und die Punkte des Bauelementes beinhaltet, die die kleinsten Z-Werte enthält. Bei üblichen Modellierungsprinzipien kann damit immer eine sinnvolle Ebene bestimmt werden. In anderen Fällen kann interaktiv eine beliebige andere Ebene bestimmt werden (siehe Abb. 56).

Wird die Bauelementbasisfläche akzeptiert, plaziert das System das Bauelement in die geometrische Mitte einer Fläche des Schaltungsträgers, die als die temporäre Positionierfläche des Schaltungsträgers selektiert wurde. Besitzt die Positionierfläche hingegen in der geometrischen Mitte eine Aussparung, wird als temporäre Position ein Position am Rand der Positionierfläche vorgeschlagen.

Die interaktive Positionierung des Bauelementes wird höchst ergonomisch unterstützt: Das Bauelement wird durch Selektion ausgewählt und durch Verschieben des Positionierhilfsmittels (vorzugsweise Maus) auf der Schaltungsträgerfläche bewegt. Solange die Selektion aufrechterhalten wird, folgt das Bauelement den Bewegungen der Maus. Die geometrische Ausprägung der Fläche spielt dabei keine Rolle. Die Positionsvariable wird dabei im lokalen u,v-Koordinatensystem der Fläche bestimmt. Die Orientierung des Bauelementes wird dabei dynamisch derart angepaßt, daß dieses sich bezüglich seiner Basisfläche normal auf die Schaltungsträgergrundfläche ausrichtet.

Das Auftreffen des Bauelementes auf einen Rand der Fläche beendet den Positioniervorgang nicht, vielmehr wird das Bauelement bei Überschreiten eines Überhangschwellwertes auf die benachbarte Fläche des Schaltungsträgers positioniert. Dabei wird ebenfalls die Flächennormale des Bauelementbasisfläche auf die Positionierfläche ausgerichtet.

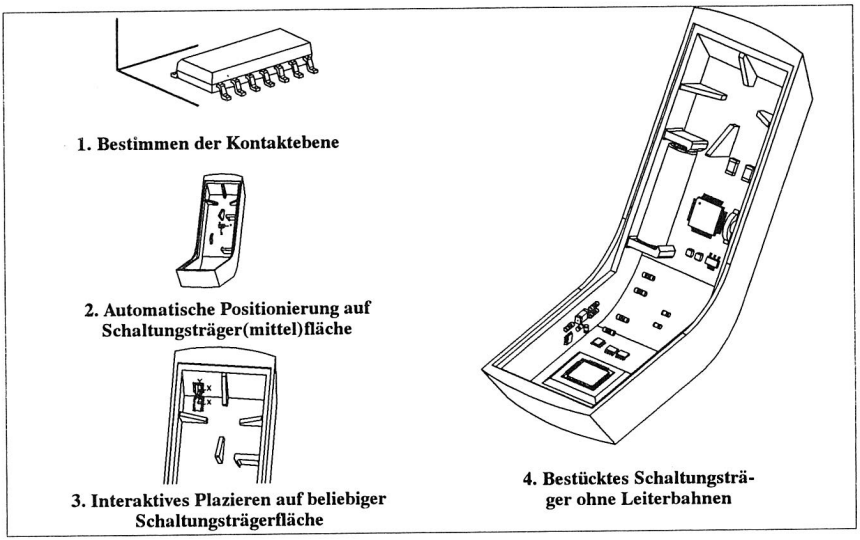


Abb. 56 Vorgehen und Ergebnis der Bauelementeplazierung

Die graphische Darstellung des Layouts und des zu plazierenden Objektes wird dynamisch reproduziert, wobei schattierte Darstellung über die Grafikanbindung zu Starbase oder Drahtdarstellung über X–Windows wählbar ist. Das Antwortzeitverhalten während der Plazierung war auf zugänglichen Workstation war auch bei komplexen Objekten ausreichend. Bei zunehmender Komplexität von Produkten könnte aber auch die Abstraktion von Bauelementen durch Quader nötig werden.

In /208/ wird in analytischen Versuchen die Haftfähigkeit und die Lötbarkeit von Bauelementen auf geneigten Bestückflächen untersucht. Es konnte nachgewiesen werden, daß ein Zusammenhang zwischen Bauelementgröße, -form und -gewicht besteht. In den im Layoutvorgang realisierten Bauelementpositionen wird jedoch auf derartige Zusammenhänge noch keine Rücksicht genommen, da im Designprozeß noch keine Aussagen über die Lage des Schaltungsträgers während des Bestück-, bzw. Lötvorganges gemacht werden kann. Eine Überprüfung dieser Abhängigkeiten kann erst während der Prozeßplanungsphase als MRC (*Manufacturing Rule Check*) realisiert werden. Hierzu gehören auch die aus der planaren Aufbautechnik bekannten MRCs zur Abstandsüberprüfung zwischen Bauelementen sowie Bauelementen und Wandungen.

7.3.3 Routing elektrischer Verbindungen auf dreidimensionaler Schaltungsträgern

Nach dem Positionieren der Bauelemente auf dem Schaltungsträger können die für die Verschaltung nötigen elektrischen Verbindungen realisiert werden.

Die aus dem elektrischen Partialmodell verfügbare Netzliste gibt die Menge der Netze und die innerhalb eines Netzes verbundenen Pins sowie Anschlußpunkte an. Neben den durch die Pins, die die zu den Bauelementbeinchen korrespondieren geben die Anschlußpunkte Punkte auf dem Schaltungsträger vor, die als direkte Kontaktpunkte zu externen Kontaktgebern fungieren. Anschaltpunkte müssen explizit auf dem Schaltungsträger identifiziert und bezeichnet werden.

Die Relation zwischen den Pins eines Bauelementes und dem geometrischen Modell wird über eine Attributierung eines künstlich erzeugten Punktes an den Bauelementbeinchen mit der Pinnummer erreicht.
Die innerhalb eines Netzes zu verlegenden Verbindungen können in einem ersten Schritt als 'Luftlinien' sichtbar gemacht werden. Dabei wird auf die physikalische Realisierbarkeit der so dargestellten Verbindungen keine Rücksicht genommen. Insbesondere werden Verbindungen unter Umständen auch innerhalb des Schaltungsträgers gezogen.
Als weitere Entflechtungsfunktionalität können die nächsten Methoden beliebig sequentiell angewandt werden:
Auf Anforderung können interaktiv identifizierte oder alle vorhandenen, als Luftlinien dargestellten Verbindungen auf die Schaltungsträgeroberfläche projieziert werden. Als Kriterium für die Projektion werden im Falle geometrischer Mehrdeutigkeiten die Leiterbahnlängen minimiert.
Durch interaktive Identifikation können dann abschnittsweise Leiterbahnen editiert werden. Dabei können folgende Maßnahmen vorgenommen werden:

- Einfügen eines neuen Abschnittsendpunktes zur Aufteilung eines Abschnittes in zwei Abschnitte.
- Repositionierung eines Abschnittsendpunktes auf der Schaltungsträgerfläche unter dynamischer Anpassung der Leiterbahnabschnitte.
- Löschen eines Abschnittsendpunktes innerhalb einer Fläche und Zusammenführung zweier Abschnitte.
- Repositionierung eines Abschnittsendpunktes auf einer Flächenkante und dynamischer Anpassung der Leiterbahnverläufe.
- Repositionierung eines Leiterbahnkreuzungspunktes.

Mit den damit verfügbaren Mechanismen muß interaktiv die Entflechtung vorgenommen werden. Als Hilfsmittel kann zu jedem Zeitpunkt die Überprüfung von Kurzschlüssen der verlegten Verbindungen bestimmt werden. Dabei werden die so verlegten Verbindungen jedoch nur als ideal linienförmige Konstrukte angenommen.

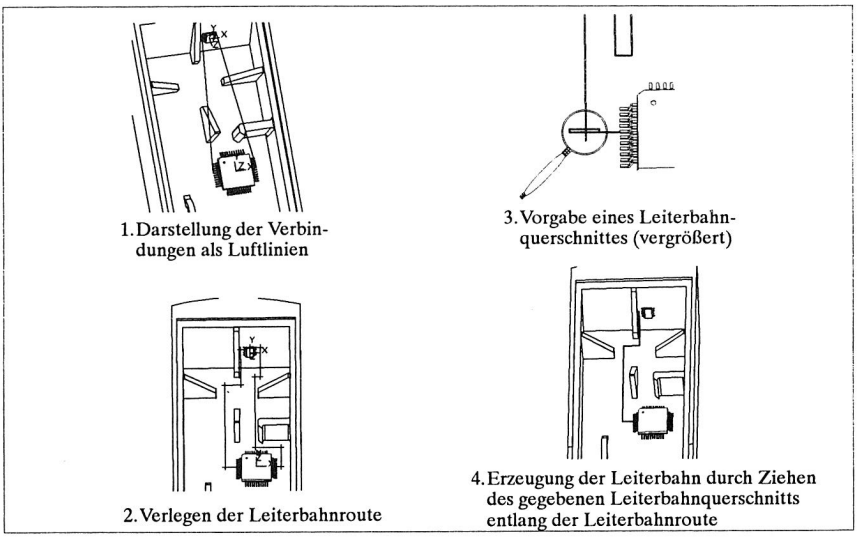


Abb. 57 Verlegen von Leiterbahnen auf dreidimensionalen Schaltungsträgern

Die Realisierung einer Leiterbahn wird dann durch Zuordnung eines abschnittsweise vorzugebenden Leiterbahnquerschnittes erreicht. Dabei sind bisher Verläufe möglich, die entlang des Abschnitts konstant, bzw. linear veränderlich sind. Dabei lassen sich beliebige Verläufe realisieren, solange ein flächiger Kontakt zwischen einzelnen Leiterbahnabschnitten erreicht wird. Mit der Spezifikation eines Leiterbahnquerschnittes wird dann durch 'Sweeping' des Querschnittsprofiles unter Vorgabe einer Transformationsfunktion ein realer voluminöser Körper erzeugt, der der realen Leiterbahn entspricht.

Die Zuordnung des Leiterbahnquerschnittes wird dabei dem Leiterbahnabschnitt als attributierte Erzeugungsvorschrift zugeordnet. Nach Vergabe der Leiterbahnquerschnittes ist jedoch weiterhin eine erneute Entflechtung möglich. Dabei wird die Repräsentation der Leiterbahn als realer Körper aufgehoben und wieder eine Linienrepräsentation angenommen. Der zugeordnete Leiterbahnquerschnitt bleibt als Attribut solange erhalten, bis der Abschnitt selbst gelöscht oder ein weiterer Abschnitt eingefügt wird.

Weiterhin ist die Repositionierung von Bauelementen möglich. Die bereits verlegten Verbindungen wechseln dabei ihre Repräsentation zu einer Liniendarstellung. Die Leiterbahnquerschnittsinformation bleibt erhalten.

Die Montage von elektronischen Bauelementen in der MID-Technologie geschieht auf ausgeformten Leiterbahnenden, den sogenannten Pads. Die Kontur die die Menge der Pads umfaßt, die benötigt wird, um ein Bauelement zu kontaktieren wird Footprint genannt. Jeder Gehäuseform kann eindeutig ein Footprint zugeordnet werden. Den aus der Bauelementebibliothek entnommenen Bauelementen kann also also über eine ein-

fache Zuordnung ein Footprint zugeordnet werden und dieses zwischen Bauelement und Schaltungsträger positioniert werden. Die Höhe der Footprint-Struktur kann dabei eingestellt werden.

7.3.4 Anwendungsbeispiel bei der Konstruktion eines ausgewählten MID-Bauteils

Als Anwendungsbeispiel wurde ein am Lehrstuhl für Fertigungsautomatisierung und Produktionssystematik entwickeltes und produziertes Beispielprodukt entworfen. Die Konstruktion des Gehäuses wurde in dem Mechanik-CAD-System Unigraphics (EDS Unigraphics) vorgenommen und über STEP-Austauschdateien in das Layoutwerkzeug eingebracht. Der Logikentwurf erfolgte in Boardstation (Mentor Graphics). Die benötigten Bauelemente sind in ACIS oder STEP-Format vorhanden.

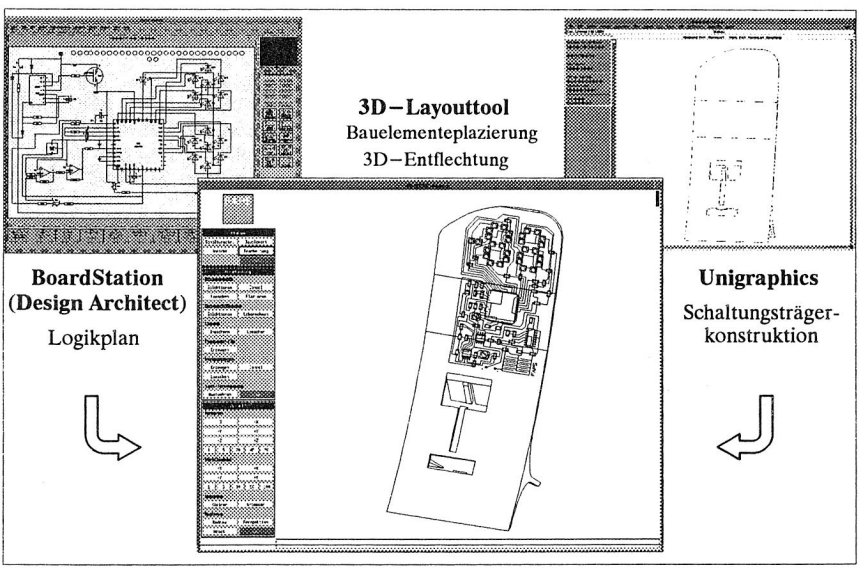


Abb. 58 CAD-Repräsentation einer MID-Komponente in dem realisierten Layoutsystem

Bei dem Beispielprodukt handelt es sich um ein Thermometer, dessen Temperaturanzeige über LEDs erfolgt.
Ziel des Beispielproduktes war die Bereitstellung einer Testumgebung für unterschiedliche Fertigungstechnologien im Bereich des Strukturierens, des Bestückens und des Lötens. Daher ergaben sich einige Einschränkungen hinsichtlich der realisierbaren geometrischen Gestalt.
Als Strukturierungstechnologie wurde die Technologie des Heißprägens sowie des Zweifachspritzgusses gegenübergestellt. Für den Heißprägevorgang eignen sich je-

doch nur bedingt gekrümmte Flächen. Daher bestehen die genutzten Strukturierungs- bzw. Bestückflächen aus drei nur leicht gekrümmten zylindrischen Flächen.

7.3.5 Anwendbarkeit und Grenzen

Mit dem realisierten CAD-Entwicklungsmodul kann die Konstruktion von MID-Komponenten effizient unterstützt werden. Das Konzept eines integrierten CAD-Systems sowohl für die mechanische als auch die elektrische Konstruktion hat sich dabei gegenüber der Kopplung von verfügbaren Systemen bewährt. Das System ist prinzipiell für alle Entwicklungsaufgaben im elektromechanischen Bereich nutzbar. Dabei ist eine Orientierung auf MID-Komponenten nur ein erster Entwicklungsschritt. Weitere Anwendungsgebiete können das Leitungsverlegungsproblem innerhalb elektrischer Geräte und Anlagen sein.

Die bestehenden Grenzen des Systems sind nur im derzeitigen Entwicklungsstand zu sehen. Weitere Anforderungen wie mechanische CAD-Funktionalität ist prinzipiell durch die Fähigkeiten des Modellierers Acis vorhanden, müssen also dem Anwender nur die Benutzerschnittstelle angeboten werden.

Die Beschränkungen im elektronischen Bereich sind durch das eingeschränkte elektronische Partialmodell sowie die darauf beschränkten Funktionen zu sehen. Als eine auch im Bereich eines Layoutmoduls wichtige Funktionalität muß dabei bisher auf das Swapping, das Neuzuordnen von Funktionen auf Bauelemente verzichtet werden. Das Entwicklungsmodul geht von einem statischen Elektronikmodell aus.

Die Leistungsfähigkeit moderner CAE-System wird durch die Leistung der Autorouter bestimmt. Obwohl leistungsfähige Autorouter auch Multilayerleiterplatten sicher entflechten, können derartige Algorithmen für das Routen auf oder in dreidimensionalen Schaltungsträgern nicht übertragen werden. Während die Layer eines Multilayerboards innerhalb der Algorithmen diskrete Werte annehmen, kann der Verlauf einer Leiterbahn innerhalb eines MID-Schaltungsträgers nahezu beliebige Werte annehmen. Die Verlaufsrestriktionen der Leiterbahn unterliegen völlig neuen Anforderungen.

Das entwickelte Werkzeug ist mit der bereitgestellten Funktion geeignet, MID-Komponenten zu entwickeln die einen Großteil der bisher möglichen Produktionstechnologie unterstützt. Die Integration von elektrischen Funktionen in komplexe Bauelemente und die Verringerung der Zahl der Bauelemente und damit der Verknüpfungen rechtfertigen den gewählten Ansatz.

8 Zusammenfassung

Ein Problem der Spezialisierung industriealisierter Wirtschaftsstrukturen in einzelne Leistungserbringer ist die Kommunikation zwischen Anbietern und Nachfragern von Gütern und Dienstleistungen. Dies gilt sowohl zwischen kooperierenden Unternehmen als auch innerhalb eines Unternehmens zwischen unterschiedlichen Abteilungen. Durch die Nutzung rechnergestützter Hilfsmittel wird diese Kommunikationsproblematik nicht gelöst, sondern auf eine neue Stufe gehoben.
Während die rechnergestützte Bearbeitung administrativer Unternehmensvorgänge oftmals zentralistisch organisiert und über Datenbanklösungen integriert wurde um Kommunikationsvorgänge zu minimieren, ließen sich technische EDV-Systeme bisher weder effizient koppeln noch zentral oder verteilt integrieren. Grund hierfür sind hochspezialisierte interne Datenstrukturen technischer Anwendungssysteme die allgemeingültigen Modellen wenig zugänglich waren.

Mit der unter der ISO definierten Norm ISO-10303 (STEP, **St**andard for the **E**xchange of **P**roduct Model Data) wird ein einheitliches Modell für produktbeschreibende Informationen geschaffen. Über die Modellbildung hinaus wird jedoch sogar eine Referenzarchitektur für die Entwicklung von Anwendungs- und Implementierungsmodellen bereitgestellt. Dabei stellt die Modellierungssprache EXPRESS fundamentalen Kern der Architektur von STEP dar.

Die Entwicklung von Werkzeugen zur Unterstützung der Implementierung der einzelnen Teile der Norm ISO-10303 stellt ein Ergebnis der geleisteten Arbeit dar. Insbesondere die Verarbeitung der Modellierungssprache EXPRESS, in der sowohl Anwendermodelle, Referenzmodelle sowie Metamodelle der Implementierungsprinzipien beschrieben werden, wurde effizient unterstützt. In der Arbeit wird auf ein flexibles Implementierungswerkzeug für jegliche in EXPRESS definierte Modelle gesetzt. Die Nutzung beliebiger EXPRESS-Modelle als Basis von Implementierungen wird dadurch möglich.

Die vier in der Norm ISO-10303 diskutierten und teilweise genormten Implementierungsmethoden Austauschdatei, Datenbasis mit Zugriffsschnittstelle SDAI (**St**andard **D**ata **A**ccess **I**nterface), Hauptspeicherstruktur und Wissensbasis wurden prototypisch in der Arbeit realisiert. In die noch laufenden Normierungstätigkeiten konnten dadurch wertvolle Spezifikationshinweise eingebracht werden.

Während zu Beginn der Entwicklung von STEP primär die Definition von Datenmodelle und daraus abgeleitet, Dateiformate zum Austausch von Produktinformationen im Vordergrund stand, stellt im weiteren Verlauf der STEP-Entwicklung die koordinierte Verwaltung von Produktinformationen einen Schwerpunkt dar. Die in Datenbanksystemen vorhandenen Fähigkeiten zur Datenverwaltung stellen einen Ansatz zur Realisierung von Produktdatenbanken dar. Die in STEP definierten Produktmodelle können dabei als Datenbankschemata benutzt werden.
Die heute in industriellen Umgebungen eingesetzten Datenbankprinzipien der relationalen und objektorientierten Datenbanksysteme werden hinsichtlich ihrer Einsetzbar-

keit als Basissysteme für STEP-Produktdatenbanken untersucht und für ein Datenbanksystem, das eine spezielle Erweiterung des relationalen Modells darstellt, eine Implementierung einer STEP-Datenbank vorgenommen.

Die primären Anwendungsprobleme, für die mit Hilfe von STEP Austauschszenarien bewältigt werden können, sind im Bereich des Datenaustausches zwischen CAD/CAE-Systemen zu finden. Besonders CAD-Systeme, die die Repräsentation dreidimensionaler technischer Objekte ermöglichen, profitieren von der ausdrucksstarken Modellbildung der STEP Geometrie-Partialmodelle. Für leistungsfähige geometrische Modellierwerkzeuge wurde ein Datenaustauschszenario skizziert und mit Hilfe von realisierten STEP-Pre- und Postprozessoren unterstützt.

Der Integrationsapekt von zukünftigen, unternehmensweiten CAD/CAE-Anwendungssystemen wird durch die Fähigkeit dieser Anwendungen bestimmt, STEP-konforme Daten aus Datenbasen zu entnehmen und für diese bereitzustellen. Während neu zu realisierende Anwendungssysteme zur Verwaltung von Produktdaten eine SDAI-Schnittstelle als integralen Bestandteil realisieren können, sind vorhandene Systeme mit Schnittstellen zu einem SDAI auszurüsten. Der geometrische Modellierer Acis wurde mit einem derartigen Interface versehen, um eine Integration von CAD-Anwendungssystemen die auf diesem Modellierer beruhen, auf STEP/SDAI-basierten Datenbasen zu ermöglichen. Die Probleme der langer Transaktionen stellen in diesem Zusammenhang neue Anforderungen Datenbankmanagementsysteme.

Die Bearbeitung von STEP-basierten Problemlösungen kann aus einer anwendungsfreien und einer anwendungsorientierten Sicht vorgenommen werden. Während Werkzeuge, die zur Unterstützung der Implementierung von Anwendungssystemen aus EXPRESS-Modellen heraus dienen, zunächst anwendungsfrei gehalten werden können, muß ein STEP-basiertes Anwendungssystem die Anforderungen, die aus dem Anwendungsreferenzmodell eines STEP-Anwendungsprotokolles stammen, inhaltlich berücksichtigen. Die aus der Entwicklungsmethodik für STEP-Anwendungsprotokolle abgeleitete problematische Beziehung zwischen Anwendungsmodell und Referenzmodell wurde kritisch hinterfragt und neue Lösungsansätze vorgestellt.

Spritzgegossene dreidimensionale Schaltungsträger stellen ein Beispiel eines mechatronischen Produktes in einer neuen Produkt- und Produktionstechnologie dar. Das Defizit von Entwicklungswerkzeugen für die Produkt- und Produktionsgestaltung dieser Produktechnologie ist vor allem durch fehlende und in den existierenden CAD/CAE-Systemen nur ineffizient realisierbare Modellbildung für derartige Produkte begründet. Das STEP-Anwendungsprotokoll 210 stellt einen akzeptablen Ansatz für eine integrierte elektromechanische Modellbildung bereit. Die in elektronisch oder mechanisch orientierten Systemen fehlenden Entwurfsfunktionen 3D-Layout und 3D-Entflechtung wurden auf einem geometrischen Modellierer realisiert. Der Prototyp wurde als integraler Bestandteil eines STEP-Anwendungssystems konzipiert und realisiert. Mit einem Anwendungsbeispiel konnte die prinzipielle Tragfähigkeit des integrierten Ansatzes nachgewiesen werden.

9 Literatur

/1/ Smith, Adam; Todd, W. B.:
An Inquiry into the Nature and Causes of the Wealth of Nations.
Reprint, Oxford, Clarendon, 1776

/2/ Feldmann, K.:
Rechnerintegrierte Montagesysteme.
Tagung Rechnerintegrierte Produktionssysteme, Universität Erlangen (1987) S. 279–294

/3/ Schneider, Stefan:
Ein STEP–orientiertes Datenbanksystem zur Integration von CAD/CAM–Anwendungen
VDI–Verlag, Düsseldorf (1993)

/4/ Owen, Jon:
STEP An Introduction
Information Geometers Ltd, Winchester (1993)

/5/ Stroustrup, Bjarne:
Die C++ Programmiersprache
Addison–Wesley, Bonn u.a.; 1992

/6/ Sedgewick, R.:
Algorithmen
Addison–Wesley, Bonn u.a.; 1991

/7/ Kernighan, Brian W.; Ritchie, Dennis M.:
Programmieren mit C.
2. Ausgabe, ANSI C. Hanser, München, Wien; 1990

/8/ Carter, Donald E.; Baker, Barbara Stilwell:
CE Concurrent Engineering The Product Development Environment for the 1990s.
Addison–Wesley, Reading; 1991

/9/ Pistor, P.:
Objektorientierung in SQL3: Stand und Entwicklungstendenzen.
In: Informatik–Spektrum, 1993, S.: 89–94

/10/ Eastman, Charles M., Fereshetian, Nirva:
Information models for use in product design: a comparison.
In: Computer–Aided Design, Volume 26, Number 7, Juli 1994; S.:551–572

/11/ Gadient, J. Anthony; Graves, Gerald, R.:
EXPRESS Driven Data Translation
In: Proceedings of the EUG, 1993

/12/ Yoo, Sang, B.; Cha, Sang K.:
Checking EXPRESS Constraints in A Heterogenous Database Environment
In: Proceedings of the EUG, 1993

/13/ Burkett, William C.:
The Semantics of Subtypes and Supertypes
In: Proceedings of the EUG, 1993

/14/ Hardwick, Martin:
Putting it all together: Using EXPRESS, CORBA and ODL to crate Open Engineering Databases
In: Proceedings of the EUG, 1993

/15/ Rando, Thomas; McCabe, Lisa:
Object-oriented Approaches to the Implementation of EXPRESS-based Systems
In: Proceedings of the EUG, 1993

/16/ Lockemann, P. C.:
Weiterentwicklung relationaler Datenbanken für objektorientierte Anwendungen
In: Informatik-Spektrum, 1993, S.: 81-88

/17/ Kemper, A.; Moerkotte, G.:
Basiskonzepte objektorientierter Datenbanksysteme
In: Informatik-Spektrum, 1993, S.: 69-80

/18/ Ahmed, Shamim; Wong, Albert; Duvvuru, Sriram; Logcher, Robert:
Object-oriented database management systems for engineering: A comparison
In: Journal of object-oriented Programming, June 1992, S.: 27-44

/19/ Lockemann, Peter C.:
Object-Oriented Database and Deductive Databases; Systems Without Market? Market Without Systems?
In: Informatik-Spektrum, 1993

/20/ Loomis, Mary E. S.:
ODBMS myths and realities
In: Journal of object-oriented Programming, July-August 1994, S.: 77-80

/21/ Hofstadter, Douglas R.:
Gödel, Escher, Bach: ein endloses geflochtenes Band
Klett-Cotta, Stuttgart, 1987

/22/ Penrose, Roger:
Computerdenken : die Debatte um künstliche Intelligenz, Bewußtsein und die Gesetze der Physik
Spektrum-der-Wissenschaft-Verlagsgesellschaft, Heidelberg, 1991

/23/ Scheer, A.-W.:
Architektur integrierter Informationssysteme: Grundlagen der Informationsmodellierung
Springer, Berlin, 1991

/24/ Feldmann, K.; Franke, J; Krebs, Th.:
OMNICAD - Integration von Funktionen zur elektrischen und mechanischen Produktgestaltung.
in: CAD '94, Paderborn 1994

/25/ Anderl, R.; Wasmer, A.:
Chancen und Risiken der Produktdatentechnologie (PDT)
in: CAD '94, Paderborn 1994

/26/ Gausemeier, Jürgen (Hrsg.):
CAD '94 - Produktdatenmodellierung und Prozessmodellierung als Grundlage neuer CAD-Systeme.
Hanser, München, 1994

/27/ Stonebraker, M.; Rowe, L.:
The Design of Postgres.
ACM SIGMOD Conf. on Management of Data, Washington D.C., USA, 1986

/28/ Senturia, S. D.; Harris, R. M.; Johnson, B. P.; Kim, S.; Nabors, K.; Shulman, M. A.; White, J. K.:
A Computer-Aided Design System for Microelectromecanical Systems (MEMCAD).
VLSI Memo 91-644, Oktober 1991, private communications

/29/ Gieling, W. F.; Suhm, A. F. (Eds.):
IMPPACT Reference Model.
ESPRIT Research Reports, Project 2165, Springer Verlag, Berlin u.a., 1992

/30/ Feldmann, K., Franke, J.:
Neue Rationalisierungspotentiale durch Funktionsintegration.
Pa Produktionsautomatisierung, 3/92, Oldenburg Verlag, München.

/31/ Feldmann, K.; Franke, J.:
New Requirements and Solutions for Product Data Processing of Three-Dimensional Molded Interconnection Devices.
Proceedings 13. IEEE International Electronics Manufacturing Technology Symposium, Baltimore, 28.–30.. Sept. 1992

/32/ Feldmann, K., Sturm, J.:
Qualitätssicherung in der Elektronikflachbaugruppenproduktion durch prozeßbegleitende Prüfung.
Tagungsband zur Leiterplatte '92, S. 223–236 VDI-Bericht 966, VDI-Verlag, Düsseldorf 1992.

/33/ Hartmann, U.; Zaderej, V.:
Aus 3D–Leiterplatten werden Produkte.
F&M, 99(1991)11, Carl Hanser Verlag, München

/34/ Ammon, P.:
Entwurf von Leiterplatten.
Hüthig Verlag, Heidelberg, 1987.

/35/ Goldhorn, B., Seyffahrt, T., Sprengel, O.:
Integrierte 3D-Konstruktion Elektronik/Mechanik.
Elektronik 11/89, S. 125–129.

/36/ IPC Molded Printed Board Subcommittee:
IPC–MB–380, Guidelines for Molded Interconnects.
Institute for Interconnecting and Packaging Electronic Circuits, Lincolnwood, Illinois, 1989

/37/ Clark, S. N.; Libes, D.:
Fed–X: The NIST Express Translator.
National Institute of Standards and Technology, 1992

/38/ Libes, D.:
Shtolo – Converting STEP Short Listings to Annotated Listings
National Institute of Standards and Technology, NISTIR 5291, 1993

/39/ Kyritsis, A.; Tzanakis, C.:
A mathematical model for the shape of wave–soldered joints on printed circuit boards.
In: Journal of Physics III France, 1993, S 1639–1658

/40/ N.N.:
ISO DIS 10303–1 Product Data Representation and Exchange – Part 1: Overview and Fundamental Principles.
TC 184/SC4

/41/ N.N.:
ISO 10303–11 Industrial automation systems and integration – Product data representation and exchange – Part 11: Description methods: The EXPRESS language reference manual
ISO, Genf, 1994

/42/ N.N.:
ISO DIS 10303–12 Product Data Representation and Exchange – Part 12: The EXPRESS–I Language Reference Manual
TC 184/SC4 N151, 31. August 1992

/43/ N.N.:
ISO 10303–21 Industrial automation systems and integration – Product data representation and exchange – Part 21: Implementation methods: Clear text encoding of the exchange structure.
ISO, Genf, 1994

/44/ N.N.:
ISO 10303–22 Industrial automation systems and integration – Product data representation and exchange – Part 22: Implementation methods: STEP Data Access Interface.
ISO TC184/SC4/WG7 N392, 1995

/45/ N.N.:
ISO 10303–23 Industrial automation systems and integration – Product data representation and exchange – Part 23: Implementation methods: C++ Programming Language Binding to the Standard Data Access Interface Specification.
ISO TC184/SC4/WG7 N393, 1995

/46/ N.N.:
ISO 10303–24 Industrial automation systems and integration – Product data representation and exchange – Part 24: Implementation methods: Standard data access interface – C language late binding
ISO TC184/SC4/WG7 N369, 1995

/47/ N.N.:
ISO 10303–26 Industrial automation systems and integration – Product data representation and exchange – Part 26: Implementation methods: Standard data access interface – IDL language binding
ISO TC184/SC4/WG7 N369, 1995

/48/ N.N.:
ISO CD 10303–31 Product Data Representation and Exchange – Part 31: Conformance Testing Methodology and Framework: General Concepts.
TC 184/SC4 N111 1992

/49/ N.N.:
ISO CD 10303–32 Product Data Representation and Exchange – Part 32: Conformance Testing Methodology and Framework: Requirements on Testing Laboratories and Clients.
TC 184/SC4 N111 1992

/50/ N.N.:
ISO 10303–41 Industrial automation systems and integration – Product data representation and exchange – Part 41: Integrated generic resources: Fundamentals of product description and support
ISO, Genf 1994

/51/ N.N.:
ISO 10303–42 Industrial automation systems and integration – Product data representation and exchange – Part 42: Integrated generic resources: Geometric and topological representations
ISO, Genf 1994

/52/ N.N.:
ISO 10303–43 Industrial automation systems and integration – Product data representation and exchange – Part 43: Integrated generic resources: Representation structures
ISO, Genf 1994

/53/ N.N.:
ISO 10303–44 Industrial automation systems and integration – Product data representation and exchange – Part 44: Integrated generic resources: Product structure configuration
ISO, Genf 1994

/54/ Swindells, Robert; Carpenter, Joe (Editors):
ISO 10303–45 Industrial automation systems and integration – Product data representation and exchange – Part 45: Integrated generic resources: Materials
ISO, TC184/SC4/WG3 N258 (P4), 1993

/55/ N.N.:
ISO 10303–46 Industrial automation systems and integration – Product data representation and exchange – Part 46: Integrated generic resources: Visual Presentation
ISO, Genf 1994

/56/ Feng, Shaw; Lampkin, Mark:
ISO 10303–47 Industrial automation systems and integration – Product data representation and exchange – Part 47: Integrated generic resources: Shape variation tolerances
ISO, TC184/SC4/WG3 N257 (P3), 1993

/57/ Dunn Mark:
ISO 10303–48 Industrial automation systems and integration – Product data representation and exchange – Part 48: Integrated generic resources: Form Features
ISO, ISO TC184/SC4/WG3, 1991

/58/ N.N:
ISO 10303–49 Industrial automation systems and integration – Product data representation and exchange – Part 49: Integrated generic resources: Process structure and properties
ISO, Genf 1995

/59/ Grout, S. (Editor):
ISO 10303–103 Industrial automation systems and integration – Product data representation and exchange – Part 103: Integrated Application Resources: Electrical Hierarchical Connectivity
ISO TC184/SC4/WG9 N74C P10–Connectivity, 1994

/60/ N.N.:
ISO 10303–201 Industrial automation systems and integration – Product data representation and exchange – Part 201: Application protocol: Explicit draughting
ISO, Genf, 1994

/61/ N.N.:
ISO 10303–203 Industrial automation systems and integration – Product data representation and exchange – Part 203: Application protocol: Configuration controlled design
ISO, Genf, 1994

/62/ Thurmann, Th. R; Summitt, D. (Eds.):
ISO 10303–210 Industrial automation systems and integration – Product data representation and exchange – Part 210: Application protocol: Printed circuit assembly product design data
ISO TC184/SC4/JWG9 N5–94 (P10), 1994

/63/ Nerke, R; Holland, M.: (Eds.):
ISO 10303–212 Industrial automation systems and integration – Product data representation and exchange – Part 212: Application protocol: Electrotechnical design and installation
ISO TC184/SC4/JWG9 N???, 1995

/64/ N.N.:
ISO DIS 10303–214 Product Data Representation and Exchange – Part 214: Application protocol: Automotive design.
TC 184/SC4 N??? 1995

/65/ Goult, Ray; (Eds.):
ISO DIS 10303–512 Product Data Representation and Exchange – Part 512: Application Interpreted Constructs: Advanced Boundary Representation.
TC 184/SC4 WG4 N606c 1994

/66/ Eigner, Martin:
Einführung und Anwendung von CAD–Systemen:
Hanser: München, Wien; 1986

/67/ Encarnaçao, J.; Schuster, R.; Vöge, E. (Eds.):
Product Data Interfaces in CAD/CAM Applications.
Springer, Berlin, Heidelberg, New York Tokyo, 1986

/68/ Encarnaçao, J.; Krause, F.–L. (Eds.):
File Structures and Data Bases for CAD.
North Holland Publishing Company, Amsterdam, New York, Oxford; 1982

/69/ Stillwell H. Richard:
Electronic Product Design for Automated Manufacturing
Marcel Dekker, New York, Basel; 1989

/70/ Bey, I.; Leuridan, J. (Eds.):
ESPRIT Project 332: CAD*I CAD Interfaces Status Report 4.
Kernforschungszentrum Karlsruhe GmbH, Karlsruhe; 1988

/71/ Requicha, Aristides A. G.:
Representations for Rigid Solids: Theory, Methods, and Systems.
Computing Surveys, Vol. 12, No. 4, 1980, S.: 437–464

/72/ Requicha, Aristides A. G.; Voelcker, H. B.:
Solid Modeling: A Historical Summary and Contemporary Assessment.
IEEE CG&A: March 1982; S. 9–24

/73/ Shah, J. J.; Mathew, A.:
Experimental investigation of the STEP Form–Feature Information Model.
computer–aided design: Volume 23 Number 4 May 1991; S.: 282–296

/74/ Mason, Tony; Brown, Doug:
lex & yacc
O'Reilly and Associates, Inc.; Sebastopol; 1990

/75/ Wilson, Peter R.:
STEP Ballot Results
IEEE Computer Graphics & Applications; May 1990, S.: 79–82

/76/ Wilson, Peter R. (Editor):
EXPRESS TOOLS and SERVICES (1990–1995)
Express Users Group Meeting 1995, Grenoble, 1995

/77/ Kroszynski, Uri I.; Palstroem, Bjarne, Trostmann, Erik; Schlechtendahl Ernst G.:
Geometric Data Transfer Between CAD Systems: Solid Models
IEEE Computer Graphics & Applications; Septermber 1989; S.: 57–71

/78/ Holland, Martin:
Das STEP–Toleranzmodell
VDI–Z 133 Nr. 10 Oktober 1991; S.: 53–56

/79/ Schreiter, Herbert; Vogel, Manfred:
Form Features Information Model (FFIM) in STEP.
edv–aspekte 2/1990 S.: 33–40

/80/ Mertens Erik:
Der Nutzen von STEP für die CAD/NC–Kopplung.
CIM–Management 3/91 S.: 37–42

/81/ Endres, Michael:
Das "CAD*I Drafting Model" – ein Datenmodell zur Beschreibung technischer Zeichnungen.
Informatik Forschung und Entwicklung; 6/1991 S.: 62–70

/82/ Grabowski, H.; Schili, B.:
Konzepte zur Implementierung genormter Schnittstellen für den Produktdatenaustausch.
Informatik Forschung und Entwicklung; 6/1991 S.: 90–101

/83/ Anderl, Reiner:
Normung von CAD–Schnittstellen.
CIM–Management 1/89 S.: 4–8

/84/ Anderl, Reiner:
Integriertes Produktmodell.
ZwF 84 (1989) 11; S.: 640–644

/85/ Reich, Thomas:
STEP und CIM OSA – verschiedene Wege zu CIM?
CIM–Management 6/91 S.: 68–70

/86/ Bloor M. S.; Owen, J.:
CAD/CAM product–data exchange: the next step.
computer–aided design: Volume 23 Number 4 May 1991; S.: 237–243

/87/ McKay, A.; Bloor M. S.; Owen, J.:
Application Protokols: a position paper
MOSES Report Series, Number 13, Januar 1994

/88/ Metzger, F.:
On the Problem of AP Interoperability
IWF, ETH Zürich, 1993

/89/ Meyer, Bertrand:
Object–oriented Software Construction.
Prentice Hall, New York; 1988

/90/ Felser, Winfried; Müller, Wolfgang:
EXPRESS–P – Extending EXPRESS for Process Modeling and Monitoring.
Proceedings of the ASME 1994, Minneapolis, MI, September 11–14, 1994

/91/ Lehrenfeld, Georg; Müller, Wolfgang; Wiechers Norbert:
Parallel Validation of STEP Files.
Proceedings of the CEEDA94, Poole, UK, April 7–8, 1994

/92/ Buijs, Frank; Lehrenfeld, Georg; Müller Wolfsgang, Stewing, Franz–Josef:
The Role of STEP in Facilitating Engineering Efforts.
CADLAB, interne Veröffentlichung, 1994

/93/ Stephanie Jo Cammarata:
An Object–Oriented Data Model for Managing Computer–Aided Design and Computer–Aided Manufacturing Data Bases.
Dissertation University of California, Los Angeles, 1986

/94/ Bussche, Jan Van den; Heuer, Andreas:
Using SQL with Object–Oriented Databases.
Informatik–Bericht 92/7, Institut für Informatik, Clausthal–Zellerfeld, September 1992

/95/ Anderl, Reiner; Castro, Pablo:
CAD/CAM: auf dem Weg zu einer branchenübergreifenden Integration.
Springer, Berlin.., 1990

/96/ McKay, Alison; Davies, Mark:
EXPRESS to C++: A mapping for the sub/supertype construct.
In: Proceedings of the EUG, 1994

/97/ McKay, Alison; Bloor, Susan M; Erens, Frederick; Owen, Jon:
EXPRESS–I–G: A graphical notation for EXPRESS–I
In: Proceedings of the EUG, 1993

/98/ Barkmeyer, Ed:
Express–to–Matisse – Tools for construction of an Object–Oriented Database fom an Express Model.
In: Proceedings of the EUG, 1994

/99/ Mak, Helium; Cloutier, Nathalie:
A "Non–Standardized" STEP Application.
In: Proceedings of the EUG, 1994

/100/ Berre Arne–Jørgen; Høgberg, Frode, :
Object–Oriented Design and Implementation of an EXPRESS–based Product Model Database.
In: Proceedings of the EUG, 1994

/101/ Griffin, C.R.:
Testing an extension of an EXPRESS data model to ensure conformance with a set of rules.
In: Proceedings of the EUG, 1994

/102/ Alt, Jochen:
Optimizing EXPRESS rule evaluation with fine–grained, dynamic materialization in CAD applications.
In: Proceedings of the EUG, 1994

/103/ Herbst, Axel:
Archiving of Data in an EXPRESS/SDAI Database.
In: Proceedings of the EUG, 1994

/104/ Loffredo, David; Hardwick, Martin:
Efficient Database Implementation of EXPRESS Information Models.
In: Proceedings of the EUG, 1994

/105/ Rando, Tom; McCabe, Lisa; Paoloni, Merilee:
Mapping EXPRESS/SDAI into the CORBA Standard.
In: Proceedings of the EUG, 1994

/106/ Sanderson, Donald B.:
A Schema Migrator for EXPRESS: Theory and Practice.
In: Proceedings of the EUG, 1994

/107/ Kendall, John M. P.:
A Formal Specification for Mapping Between Graphical Modelling Languages.
In: Proceedings of the EUG, 1994

/108/ Bailey, I.D.:
Latest Thinking in EXPRESS–M.
In: Proceedings of the EUG, 1994

/109/ Zhou, Mingwei; Bloor, Susan; de Pennington, Alan:
Towards an EXPRESS Oriented Executable Language for the Rapid Prototyping of Product Data Models.
In: Proceedings of the EUG, 1994

/110/ Müller Wolfgang, Felser, Winfried; Buijs, Frank:
EXPRESS Meets Process Modeling.
In: Proceedings of the EUG, 1994

/111/ Bühlmann, Thomas:
"Beitrag zur EUG 94"
In: Proceedings of the EUG, 1994

/112/ Hardwick, Martin:
Mapping AIMs to ARMs: Three Approaches.
In: Proceedings of the EUG, 1994

/113/ Schönefeld, F.; Staub, G.; Maier, M.; Musto, A.:
Multiple Class Membership and Supertype Constraint Handling – Concepts and Implementation Aspects.
In: Proceedings of the EUG, 1994

/114/ Dach, Miroslav; Hoimyr, Nils; Saarela, Janne; Vuoskoski, Jouko:
Using EXPRESS in High Energy Physics Research Environment.
In: Proceedings of the EUG, 1994

/115/ Gonçalves, Ricardo Jardim:
Industrial Integration Problems – Can STEP Help?
In: Proceedings of the EUG, 1994

/116/ Wagner, Manfred:
Der STEPIntegrator als Datendrehscheibe für die Integration von CAD/CAM–Systemen.
Siemens interne Veröffentlichung, Nürnberg, 1994

/117/ Nill, Roland; Fischer, Manfred; Wenzel, Bernd:
STEP–basierte Datenbanken.
CAD–CAM Report, Nr.12, 12. Jahrgang, Dezember 1993, S. 84 – 91

/118/ Hellmuth, Thomas W.; Maier, Markus:
STEP: Chance oder Werbeslogan.
CAD–CAM Report, Nr.10, 13. Jahrgang, Oktober 1994, S. 44 – 55

/119/ Lorenz, Hans–Peter; Lutz Richard:
STEP: Datenaustausch bei Robotersystemen.
CAD–CAM Report, Nr.10, 13. Jahrgang, Oktober 1994, S. 56 – 62

/120/ Price, David; McCabe, Lisa; Rando, Tom (Eds.):
Standard Data Access Interface.
ISO TC184/SC4/WG7 N370, 15. November 1994

/121/ Palmer, Mark; Gilbert, Mitch (Eds.):
Guidelines for the development and approval of STEP application protocols.
ISO TC184/SC4 PMAG N103, 30. November 1993

/122/ The Prostgres Group (Eds.):
The Postgres User Manual.
University of California at Berkeley, Computer Science Div., Dept of EECS

/123/ The Prostgres Group (Eds.):
The Postgres Reference Manual.
University of California at Berkeley, Computer Science Div., Dept of EECS

/124/ Machner, Bodo:
Die Anwendung von EXPRESS–Spezifikationen für den konzeptionellen Datenbankentwurf.
Institut für Informatik und Rechentechnik, Berlin, 1991

/125/ Klement, Kornél:
Präsentation mit STEP.
Springer, Berlin u.a., 1992

/126/ Wilson, Peter:
PDES STEPs Forward
IEEE CG&A: March 1989; S. 79–80

/127/ Wilson, Peter:
Information Modeling
IEEE CG&A: December 1987; S. 65–67

/128/ Wilson, Peter:
Information and/or Data
IEEE CG&A: November 1987; S. 58–61

/129/ Wilson, Peter:
Standards: Past Tense and Future Perfect?
IEEE CG&A: January 1991; S. 44–48

/130/ Hurson, A.R.; Simin, H. Pakzad:
Evolution and Performance Issues
IEEE Computer: February 1993; S. 48–60

/131/ Krause, F.–L.; Kimura, F.; Kjellberg, T.; Lu, S. C.–Y.:
Product Modeling
Annals of the CIRP, 1993

/132/ ElMaraghy, Hoda A.:
Evolution and Future Perspectives of CAPP
Annals of the CIRP, Vol. 42/2/1993

/133/ Kramer, Thomas R.:
Extracting STEP Geometry and Topology from a Solid Modeler: Parasolid–to–STEP
NIST, NISTIR 4577, 1991

/134/ Oehlmann Roland:
MARITIME Rules for Subsetting a Multiple Schema EXPRESS Model
BIBA, 12. April 1995

/135/ J.–P. Tremblay, P. G. Sorenson:
The Theory and Practice of Compiler Writing
McGraw–Hill, New York e.al; 1987

/136/ A. V. Aho, J. D. Ullman
Principles of Compiler Design
Addison–Wesley, Reading, Massachusets e.al.; 1978

/137/ H. Herold
Lex und Yacc –Lexikalische und syntaktische Analyse
Addison–Wesley, Bonn u.a.; 1992

/138/ J. Hoschek, D. Lasser:
Grundlagen der geometrischen Datenverarbeitung.
Teubner, Stuttgart, 1989

/139/ M. Mäntylä:
An Introduction to Solid Modeling.
Computer Science Press, Rockville, Maryland, 1988

/140/ J.–F. Grätz:
Handbuch der 3D–CAD–Technik.
Siemens, Berlin, 1989

/141/ Warnecke, G.; Radtke, M.; Filser, F.:
Unternehmens– und prozeßspezifische Produktmodelle – Produktmodelle als Grundlage vernetzter Produktentwicklungsprozesse.
wt–Produktion und Management 85 (1995), Springer Verlag, S. 132–136

/142/ Staub, G.; Nieva, A.; Scheder, H.; Velten, V.:
ESPRIT Projekt PISA – Ergebnisse und Empfehlungen für die Produktdatentechnologie.
Produktdatenjournal 1/1995 2. Jahrgang, Dressler Verlag, Heidelberg, 1995

/143/ Holland, M.; Machner, B.:
Produktdatenmanagement auf der Basis von ISO 10303–STEP
In: CIM Management 11 1995(4), GITO–Verlag, S. 32–40

/144/ Baumann, M.:
Anwendungsspezifische Erweiterung von Konstruktionssystemen für geometrisch–gestalterische Tätigkeiten unter Berücksichtigung einer systemneutralen Datenhaltung
Dissertation, Aachen, Technische Hochschule, Shaker Verlag

/145/ Berre, A.–J.; Oldevik, J.:
EXPRESS–OODL – Object–oriented externsion to EXPRESS as a basis for Information Interchange, Sharing and Systems Interoperability.
Express Users Group 1995, Grenoble, 1995

/146/ Zhao, R.; Müller, W.; Kaufmann, H.–J.; Kern, T; Buijs, F.:
An Editor for the Rapid Prototyping of EXPRESS–G Models
Express Users Group 1995, Grenoble, 1995

/147/ Krebs. Th.:
EXPRESS2KAPPA – Translating EXPRESS Into a CASE–Tool
Express Users Group 1995, Grenoble, 1995

/148/ Krebs. Th.:
Translating EXPRESS Models to the Extended Relational Database Management System POSTGRES
Express Users Group 1995, Grenoble, 1995

/149/ Libes, D.; Clark, S. N.:
An Object–Oriented Tcl/Tk Binding for the Interpreted Control of the NIST EXPRESS Toolkit in the NIST STEP Application Protocol Development Environment.
Express Users Group 1995, Grenoble, 1995

/150/ Matthews, B.; Bicarregui, J.:
Process Modelling in Control Systems Design
Express Users Group 1995, Grenoble, 1995

/151/ Ait–Ameur, Y.; Pierra, G.; Sardet, E.:
Using the EXPRESS language for metaprogramming
Express Users Group 1995, Grenoble, 1995

/152/ Su, S. Y. W.; Lam, H.; Yu, T.–F.; Lee, S.; Arroyo, J.:
On Bridging and Extending OMG/IDL and STEP/EXPRESS for SAchieving Information Sharing and Systems Interoperability
Express Users Group 1995, Grenoble, 1995

/153/ Staub, G.; Maier, M.:
Object Modelling Technique (OMT) and EXPRESS – Comparision of "Two Worlds"
Express Users Group 1995, Grenoble, 1995

/154/ Ait–Ameur, Y.; Pierra, G.; Sardet, E.:
Extending the modelling power of the EXPRESS language by formal annotations
Express Users Group 1995, Grenoble, 1995

/155/ Bicarregui, J.; Matthews, B.:
Integrating EXPRESS and SGML for Document Modelling in Control Systems Design
Express Users Group 1995, Grenoble, 1995

/156/ Sanderson, D. B.; Marks, J. T.; Tolbert, R.:
EXPRESS in Quality Monitoring
Express Users Group 1995, Grenoble, 1995

/157/ Jardim–Gonçalves, R.; Vital, M.; Sousa, P.; Silva, H. Steiger–Garção, A.:
Using SIP in Industrial Environments – Achieved Results and Experiances
Express Users Group 1995, Grenoble, 1995

/158/ Jacolot, Ch.:
Modelling GDMO in EXPRESS
Express Users Group 1995, Grenoble, 1995

/159/ Han, S.–H.; Shin, Y.–J.:
Exchange of CAD Data for Ship Designs by the STEP Standard
Express Users Group 1995, Grenoble, 1995

/160/ Kieckenbeck, J.; Siegenthaler, A.; Schlageter, G.:
EXPRESS to C++: A mapping of the Type–System
Express Users Group 1995, Grenoble, 1995

/161/ Sauder, D.; Morris, K.C.:
Design of a C++ Software Library for Implementing Express
Express Users Group 1995, Grenoble, 1995

/162/ Metzger F.:
– EXPRESS Implementation Theory – Complex Instances & Fast Attribute Access
Express Users Group 1995, Grenoble, 1995

/163/ Kretzberg, Th.; Wilkes, W.:
EXPRESS+ und SDAI+ Specification and Automatic Derivation of Higher Level Programming Interfaces
Express Users Group 1995, Grenoble, 1995

/164/ Sauter, G.; Käfer, W.:
EXPRESS as the Common Data Model in Federated Database Systems
Express Users Group 1995, Grenoble, 1995

/165/ Nakamura, I.; Kawabata, S.; Yokota, H.; Kojima, T.; Kimura, F.:
An EXPRESS–based Data Repository Manager for the STEP–based Data Exchange Software
Express Users Group 1995, Grenoble, 1995

/166/ Scholz, G.; Wilkes, W.:
Using EXPRESS for the Formal Specification of Mappings between EXPRESS Models
Express Users Group 1995, Grenoble, 1995

/167/ Bailey, I. D.; Birring, G. S.; Mead, M.:
A Meta Model for EXPRESS–M
Express Users Group 1995, Grenoble, 1995

/168/ Sanderson, D. B.:
Semantic Effects of Migration Operations on EXPRESS Schemas
Express Users Group 1995, Grenoble, 1995

/169/ Hardwick, M.:
Lessons Learned Mapping the AP203 AIM to the AP203 ARM Using EXPRESS–V
Express Users Group 1995, Grenoble, 1995

/170/ Liebich, Th.; Amor, R.; Verhoef, M.:
A Survey of Mapping Methods available within the Product Modelling Arena
Express Users Group 1995, Grenoble, 1995

/171/ Mohrmann, J.; Speck, H.–J.:
Das Produktmodell als Integrationsplattform für Prozeßketten
in: CAD '94, Paderborn 1994

/172/ Palmer, M.; Gilbert, M. (Editors):
Guidelines for the development and approval of STEP application protocols, version 1.1
ISO TC 184/SC4 PMAG N103, 1993

/173/ Feeney, A. B.; Craig, D. (Editors):
Guidelines for the development of mapping tables
ISO TC 184/SC4/WG4 N507, 1995

/174/ Gilbert, M.; Yang, Y. (Editors):
Guidelines for AIM Development
ISO TC 184/SC4/WG4 N302, 1993

/175/ Holland, M; Besekau, J. (Editors):
Interoperability of Application Protocols
Arbeitspapier des ProSTEP IO–Arbeitskreises, Darmstadt, 1995

/176/ Foley, J. D.; van Dam, A.; Feiner, S. K.; Hughes, J. F.; Phillips, R. L.:
Grundlagen der Computergraphik – Einführung, Konzepte Methoden
Addison–Wesley, Bonn; Paris; Reading, Mass. [u.a.], 1994

/177/ Neider, J.; Davis, T.; Woo, M.:
OpenGL Programming Guide
Addison–Wesley, Reading, Mass. [u.a.], 1995

/178/ Rogelberg, D. (Editor):
OpenGL Reference Manual
Addison–Wesley, Reading, Mass. [u.a.], 1995

/179/ Giehling, W.:
ESPRIT project 6876 PISA – Platform for Information Sharing by CIME Applications – Overview
In: European Product Data Technology Days '95, S. 115–122

/180/ Carl, B.:
Datenbankorientierte Kopplung von CAD–Moduln – Ein Beitrag zur Lösung des Interface–Problems im CAD–Bereich
Dissertation, Aachen, 1989

/181/ Wedekind, H.:
Protogeometrische Deutung von STEP–Verfahrensketten
Informatik–Spektrum 15 (1992), S. 167–168

/182/ Wedekind, H.:
Are the Terms "Version" and "Variant" Orthogonal to One Another?
Sigmod Record, Vol 23, No. 4, December 1994, S. 3–7

/183/ Cameron, D.; Rosenblatt, B.:
Learning GNU Emacs
O'Reilly & Associates, Sebastotol, 1991

/184/ Ort, A.; Bugow, R.:
Die Schaffung eines offenen Teilebibliothekkonzeptes
Institut für Maschinenwesen der Technischen Universität Clausthal, Institutsmitteilung Nr. 19, 1994

/185/ Bugow, R.; Ort, A.:
PLUS: Die Bereitstellung von Norm–, Wiederhol– und Zukaufteilen für die rechnerunterstützte Konstruktion
Institut für Maschinenwesen der Technischen Universität Clausthal, Institutsmitteilung Nr. 18, 1993

/186/ Lührsen, H.:
Die Entwicklung von Datenbanken für das Produktmodell der ISO–Norm STEP.
Dissertation, Erlangen, 1996

/187/ Eggers, J.:
Implementing EXPRESS in SQL
ISO TC 184/SC4/WG1 N292, St. Louis

/188/ Morris, K.C.:
Translating EXPRESS to SQL: A Users Guide
NIST IR 4341, Gaithersburg 1990

/189/ Schenck, Douglas A.; Wilson, Peter R.:
Information Modeling – The EXPRESS Way.
Oxford University Press, Oxford u.a., 1994

/190/ Hardwick, Martin; Spooner, David L.; Morris, K. C.:
Sharing Manufacturing Information in Virtual Enterprises
In: Communications of the ACM, Vol. 39, No.2, 1996

/191/ Object Management Group:
IDL C++ Language Mapping Specification.
OMG Dokument 94–9–14, 1994

/192/ Object Management Group:
The Common Object Request Broker.
OMG Dokument 94–12–29, 1993

/193/ IntelliCorp Inc.
Kappa Application System Reference, Version 3.0
Intellicorp Inc. 1993

/194/ Wedekind, Hartmut:
Datenbanksysteme I
Blibliographisches Institut, Mannheim u.a., 1981

/195/ Wenzel, Bernd G.; Müllenbach, Sabine:
EXPRESS Version2 – Requirements and Project Proposal
Digital Equipment Corporation, Maynard, Massachusetts, 1992

/196/ Spiby, Phil; Owen, Jon:
Semantic Meta-model for next edition of EXPRESS
ISO TC184/SC4/WG5 N253 (P2), 1996

/197/ Staub, Günther; Maier, Markus:
ECCO Tool Kit – User Reference Manual
Institut für Rechneranwendung in Planung und Konstruktion (RPK), Karlsruhe, 1995

/198/ Heuer, Andreas:
Objektorientierte Datenbanken – Konzepte, Modelle, Systeme.
Addison-Wesley, Bonn, u.a., 1992

/199/ Atkinson, Malcom; Bancilhon, François; DeWitt, David; Dittrich, Klaus; Maier, David; Zdonik, Stanley:
The Object-Oriented Database System Manifesto
In: Deductive and Object-Oriented Databases, Elsevier Science Publishers B.V. (North-Holland), 1990

/200/ Codd, Edgar F; Rustig, R.:
Further Normalization of the database relational model
In: Courant Computer Science Symposium "Data Base Systems", Prentice Hall, New York 1971, S. 33-64

/201/ McLay, Michael; Morris, Katherine C.:
The NIST STEP Class Library (STEP Into The Future)
National Institute of Standards and Technology, NISTIR 4411, 1990

/202/ Shape Data :
Parasolid v5.0 – Functional Description
Cambridge, 1992

/203/ Shape Data :
Parasolid v5.0 – Programming Reference Manual
Cambridge, 1992

/204/ Feldmann, K.; Brand, A.; Franke, J.:
Räumliche spritzgegossene Schaltungsträger herlfen Kosten sparen.
PRONIC (1993), Vogel Verlag Würzburg, 1993

/205/ IPC Molded Printed Board Subcommitee IPC-MB-380,
Guideleines for Molded Interconnects.
Institute for Interconnecting and Packaging Electronic Circuits, Lincolnwood, Illinois, 1989

/206/ N.N.:
Information Technology – Database Languages – SQL
ISO/IEC 9075:1992, Genf, 1992

/207/ Weber, R.:
SQL2-Norm und SQL3-Projekt
in: Informatik-Spektrum 16 (1993) 2, S. 95

/208/ Gerhard, M.:
Löten von MID's
In: MID '94 – 1. International Congress on Moulded Interconnect Devices, Erlangen, 1994

/209/ ANSI/X3/SPARC Study Group on Data Base Management Systems
Interim Report 75-02-08, in: FDT. Bulletin of the ACM SIGMOD, 7 (1975) 2

/210/ Spatial Technology Inc.
Acis Geometric Modeler – Application Guide, Version 1.7
Spatial Technology Inc., Boulder, 1995

Lebenslauf

	Thomas Krebs
	geboren am 15.08.1964 in Schwabach
1970 – 1974	Grundschule in Schwabach
1974 – 1983	Adam-Kraft-Gymnasium in Schwabach, Abitur
1983 – 1984	Grundwehrdienst
1984 – 1990	Studium der Fertigungstechnik an der Friedrich-Alexander-Universität Erlangen-Nürnberg
1990 – 1996	Wissenschaftlicher Mitarbeiter am Lehrstuhl für Fertigungsautomatisierung und Produktionssystematik an der Friedrich-Alexander-Universität Erlangen-Nürnberg Leiter: Prof. Dr.-Ing. K. Feldmann

Reihe Fertigungstechnik Erlangen

Band 1
Andreas Hemberger
Innovationspotentiale in der rechnerintegrierten Produktion durch wissensbasierte Systeme
208 Seiten, 107 Bilder. 1988. Kartoniert.

Band 2
Detlef Classe
Beitrag zur Steigerung der Flexibilität automatisierter Montagesysteme durch Sensorintegration und erweiterte Steuerungskonzepte
194 Seiten, 70 Bilder. 1988. Kartoniert.

Band 3
Friedrich-Wilhelm Nolting
Projektierung von Montagesystemen
201 Seiten, 107 Bilder, 1 Tabelle. 1989.
Kartoniert.

Band 4
Karsten Schlüter
Nutzungsgradsteigerung von Montagesystemen durch den Einsatz der Simulationstechnik
177 Seiten, 97 Bilder. 1989. Kartoniert.

Band 5
Shir-Kuan Lin
Aufbau von Modellen zur Lageregelung von Industrierobotern
168 Seiten, 46 Bilder. 1989. Kartoniert.

Band 6
Rudolf Nuss
Untersuchungen zur Bearbeitungsqualität im Fertigungssystem Laserstrahlschneiden
206 Seiten, 115 Bilder, 6 Tabellen. 1989. Kartoniert.

Band 7
Wolfgang Scholz
Modell zur datenbankgestützten Planung automatisierter Montageanlagen
194 Seiten, 89 Bilder. 1989. Kartoniert.

Band 8
Hans-Jürgen Wißmeier
Beitrag zur Beurteilung des Bruchverhaltens von Hartmetall-Fließpreßmatrizen
179 Seiten, 99 Bilder, 9 Tabellen. 1989. Kartoniert.

Band 9
Rainer Eisele
Konzeption und Wirtschaftlichkeit von Planungssystemen in der Produktion
183 Seiten, 86 Bilder. 1990. Kartoniert.

Band 10
Rolf Pfeiffer
Technologisch orientierte Montageplanung am Beispiel der Schraubtechnik
216 Seiten, 102 Bilder, 16 Tabellen. 1990. Kartoniert.

Band 11
Herbert Fischer
Verteilte Planungssysteme zur Flexibilitätssteigerung der rechnerintegrierten Teilefertigung
201 Seiten, 82 Bilder. 1990. Kartoniert.

Band 12
Gerhard Kleineidam
CAD/CAP: Rechnergestützte Montagefeinplanung
203 Seiten, 107 Bilder. 1990. Kartoniert.

Band 13
Frank Vollertsen
Pulvermetallurgische Verarbeitung eines übereutektoiden verschleißfesten Stahls
XIII + 217 Seiten, 67 Bilder, 34 Tabellen. 1990. Kartoniert.

Band 14
Stephan Biermann
Untersuchungen zur Anlagen- und Prozeßdiagnostik für das Schneiden mit CO_2-Hochleistungslasern
VIII + 170 Seiten, 93 Bilder, 4 Tabellen. 1991. Kartoniert.

Band 15
Uwe Geißler
Material- und Datenfluß in einer flexiblen Blechbearbeitungszelle
124 Seiten, 41 Bilder, 7 Tabellen. 1991. Kartoniert.

Band 16
Frank Oswald Hake
Entwicklung eines rechnergestützten Diagnosesystems für automatisierte Montagezellen
XIV + 166 Seiten, 77 Bilder. 1991. Kartoniert.

Band 17
Herbert Reichel
Optimierung der Werkzeugbereitstellung durch rechnergestützte Arbeitsfolgenbestimmung
198 Seiten, 73 Bilder, 2 Tabellen. 1991. Kartoniert.

Band 18
Josef Scheller
Modellierung und Einsatz von Softwaresystemen für rechnergeführte Montagezellen
198 Seiten, 65 Bilder. 1991. Kartoniert.

Band 19
Arnold vom Ende
Untersuchungen zum Biegeumformen mit elastischer Matrize
166 Seiten, 55 Bilder, 13 Tabellen. 1991. Kartoniert.

Band 20
Joachim Schmid
Beitrag zum automatisierten Bearbeiten von Keramikguß mit Industrierobotern
XIV + 176 Seiten, 111 Bilder, 6 Tabellen. 1991. Kartoniert.

Band 21
Egon Sommer
Multiprozessorsteuerung für kooperierende Industrieroboter in Montagezellen
188 Seiten, 102 Bilder. 1991. Kartoniert.

Band 22
Georg Geyer
Entwicklung problemspezifischer Verfahrensketten in der Montage
192 Seiten, 112 Bilder. 1991. Kartoniert.

Band 23
Rainer Flohr
Beitrag zur optimalen Verbindungstechnik in der Oberflächenmontage (SMT)
186 Seiten, 79 Bilder. 1991. Kartoniert.

Band 24
Alfons Rief
Untersuchungen zur Verfahrensfolge Laserstrahlschneiden und -schweißen in der Rohkarosseriefertigung
VI + 145 Seiten, 58 Bilder, 5 Tabellen. 1991. Kartoniert.

Band 25
Christoph Thim
Rechnerunterstützte Optimierung von Materialflußstrukturen in der Elektronikmontage durch Simulation
188 Seiten, 74 Bilder. 1992. Kartoniert.

Band 26
Roland Müller
CO_2-Laserstrahlschneiden von kurzglasverstärkten Verbundwerkstoffen
141 Seiten, 107 Bilder, 4 Tabellen. 1992. Kartoniert.

Band 27
Günther Schäfer
Integrierte Informationsverarbeitung bei der Montageplanung
195 Seiten, 76 Bilder. 1992. Kartoniert.

Band 28
Martin Hoffmann
Entwicklung einer CAD/CAM-Prozeßkette für die Herstellung von Blechbiegeteilen
149 Seiten, 89 Bilder. 1992. Kartoniert.

Band 29
Peter Hoffmann
Verfahrensfolge Laserstrahlschneiden und -schweißen : Prozeßführung und Systemtechnik in der 3D-Laserstrahlbearbeitung von Blechformteilen
186 Seiten, 92 Bilder, 10 Tabellen. 1992. Kartoniert.

Band 30
Olaf Schrödel
Flexible Werkstattsteuerung mit objektorientierten Softwarestrukturen
180 Seiten, 84 Bilder. 1992. Kartoniert.

Band 31
Hubert Reinisch
Planungs- und Steuerungswerkzeuge zur impliziten Geräteprogrammierung in Roboterzellen
XI + 212 Seiten, 112 Bilder. 1992. Kartoniert.

Band 32
Brigitte Bärnreuther
Ein Beitrag zur Bewertung des Kommunikationsverhaltens von Automatisierungsgeräten in flexiblen Produktionszellen
XI + 179 Seiten, 71 Bilder. 1992. Kartoniert.

Band 33
Joachim Hutfless
Laserstrahlregelung und Optikdiagnostik in der Strahlführung einer CO_2-Hochleistungslaseranlage
175 Seiten, 70 Bilder, 17 Tabellen. 1993. Kartoniert.

Band 34
Uwe Günzel
Entwicklung und Einsatz eines Simulationsverfahrens für operative und strategische Probleme der Produktionsplanung und -steuerung
XIV + 170 Seiten, 66 Bilder, 5 Tabellen. 1993. Kartoniert.

Band 35
Bertram Ehmann
Operatives Fertigungscontrolling durch Optimierung auftragsbezogener Bearbeitungsabläufe in der Elektronikfertigung
XV + 167 Seiten, 114 Bilder. 1993. Kartoniert.

Band 36
Harald Kolléra
Entwicklung eines benutzerorientierten Werkstattprogrammiersystems für das Laserstrahlschneiden
129 Seiten, 66 Bilder, 1 Tabelle. 1993. Kartoniert.

Band 37
Stephanie Abels
Modellierung und Optimierung von Montageanlagen in einem integrierten Simulationssystem
188 Seiten, 88 Bilder. 1993. Kartoniert.

Band 38
Robert Schmidt-Hebbel
Laserstrahlbohren durchflußbestimmender Durchgangslöcher
145 Seiten, 63 Bilder, 11 Tabellen. 1993. Kartoniert.

Band 39
Norbert Lutz
Oberflächenfeinbearbeitung keramischer Werkstoffe mit XeCl-Excimerlaserstrahlung
187 Seiten, 98 Bilder, 29 Tabellen. 1994. Kartoniert.

Band 40
Konrad Grampp
Rechnerunterstützung bei Test und Schulung an Steuerungssoftware von SMD-Bestücklinien
178 Seiten, 88 Bilder. 1995. Kartoniert.

Band 41
Martin Koch
Wissensbasierte Unterstützung der Angebotsbearbeitung in der Investitionsgüterindustrie
169 Seiten, 68 Bilder. 1995. Kartoniert.

Band 42
Armin Gropp
Anlagen- und Prozeßdiagnostik beim Schneiden mit einem gepulsten Nd:YAG-Laser
160 Seiten, 88 Bilder, 7 Tabellen. 1995. Kartoniert.

Band 43
Werner Heckel
Optische 3D-Konturerfassung und on-line Biegewinkelmessung mit dem Lichtschnittverfahren
149 Seiten, 43 Bilder, 11 Tabellen. 1995. Kartoniert.

Band 44
Armin Rothhaupt
Modulares Planungssystem zur Optimierung der Elektronikfertigung
180 Seiten, 101 Bilder. 1995. Kartoniert.

Band 45
Bernd Zöllner
Adaptive Diagnose in der Elektronikproduktion
195 Seiten, 74 Bilder, 3 Tabellen. 1995. Kartoniert.

Band 46
Bodo Vormann
Beitrag zur automatisierten Handhabungsplanung komplexer Blechbiegeteile
126 Seiten, 89 Bilder, 3 Tabellen. 1995. Kartoniert.

Band 47
Peter Schnepf
Zielkostenorientierte Montageplanung
144 Seiten, 75 Bilder. 1995. Kartoniert.

Band 48
Rainer Klotzbücher
Konzept zur rechnerintegrierten Materialversorgung in flexiblen Fertigungssystemen
156 Seiten, 62 Bilder. 1995. Kartoniert.

Band 49
Wolfgang Greska
Wissensbasierte Analyse und Klassifizierung von Blechteilen
144 Seiten, 96 Bilder. 1995. Kartoniert.

Band 50
Jörg Franke
Integrierte Entwicklung neuer Produkt- und Produktionstechnologien für räumliche spritzgegossene Schaltungsträger (3-D MID)
196 Seiten, 86 Bilder, 4 Tabellen. 1995. Kartoniert.

Band 51
Franz-Josef Zeller
Sensorplanung und schnelle Sensorregelung für Industrieroboter
190 Seiten, 102 Bilder, 9 Tabellen. 1995. Kartoniert.

Band 52
Michael Solvie
Zeitbehandlung und Multimedia-Unterstützung in Feldkommunikationssystemen
200 Seiten, 87 Bilder, 35 Tabellen. 1996. Kartoniert.

Band 53
Robert Hopperdietzel
Reengineering in der Elektro- und Elektronikindustrie
180 Seiten, 109 Bilder, 1 Tabelle. 1996. Kartoniert.

Band 54
Thomas Rebhahn
Beitrag zur Mikromaterialbearbeitung mit Excimerlasern – Systemkomponenten und Verfahrensoptimierungen
148 Seiten, 61 Bilder, 10 Tabellen. 1996. Kartoniert.

Band 55
Henning Hanebuth
Laserstrahlhartlöten mit Zweistrahltechnik
157 Seiten, 58 Bilder, 11 Tabellen. 1996. Kartoniert.

Band 56
Uwe Schönherr
Steuerung und Sensordatenintegration für flexible Fertigungszellen mit kooperierenden Robotern
188 Seiten, 116 Bilder, 3 Tabellen. 1996. Kartoniert.

Band 57
Stefan Holzer
Berührungslose Formgebung mit Laserstrahlung
162 Seiten, 69 Bilder, 11 Tabellen. 1996. Kartoniert.

Band 58
Markus Schulz
Fertigungsqualität beim 3D-Laserstrahlschneiden von Blechformteilen
165 Seiten, 88 Bilder, 9 Tabellen. 1996. Kartoniert.

Band 59
Thomas Krebs
Integration elektromechanischer CA-Anwendungen über einem STEP-Produktmodell
198 Seiten, 58 Bilder, 8 Tabellen. 1996. Kartoniert.