KfK 4912
November 1991

Über Produktbeschreibungssprachen in CIM-Systemen und die Generierung zugehöriger Übersetzer

V. Dobrowolny
Institut für Reaktorentwicklung

Kernforschungszentrum Karlsruhe

KERNFORSCHUNGSZENTRUM KARLSRUHE

Institut für Reaktorentwicklung

KfK 4912

Über Produktbeschreibungssprachen in CIM-Systemen und die Generierung zugehöriger Übersetzer

Volker Dobrowolny

Kernforschungszentrum Karlsruhe GmbH, Karlsruhe

Als Manuskript gedruckt

Kernforschungszentrum Karlsruhe GmbH
Postfach 3640, 7500 Karlsruhe 1

ISSN 0303-4003

Zusammenfassung

Schwerpunkt der Arbeit ist ein objektorientiertes Konzept zur Beschreibung integrationsfähiger Datenmodelle unter Berücksichtigung möglicher Zwangsbedingungen. Dazu werden zunächst einige Anforderungen zusammengestellt, die sich bei einer integrierten Verarbeitung von Produktdatenmodellen ergeben. Es folgt eine Erörterung der Grundstrukturen von Produkten und Datenmodellen und deren wechselseitigen Zusammenhangs. Ein wirksames Hilfsmittel bildet eine grafische Beschreibungssprache ("pUR-Modellsprache"), die trotz ihres einfachen Aufbaus eine große Mächtigkeit und hohe Flexibilität aufweist. Aus der Struktur der Modellbeschreibung ergeben sich Konsequenzen für den Bau von Modellübersetzern. Zur Veranschaulichung der allgemeinen Aussagen dienen die Strukturen von zwei Kinematikmodellsprachen NIRO und ROBOT. Mengenoperationen und Prädikate bieten ein Grundgerüst , um etwa Strukturänderungen der Objekte oder funktionelle Abhängigkeiten von Attributen zu beschreiben.

About Product Description Languages in CIM Systems and the Generation of Associated Translators

Abstract

The main issue of this work is an object-oriented concept for the description of data models suitable for integration with consideration of possible constraints. First, the relevant requirements are summarized which result from an integration of product model processing. Then we discuss the basic structures of products and data models and their interrelationships. An efficient tool is the graphical representation languange ("pUR models) which - despite its simplicity - is very powerful and flexible. The structure of the model description results in consequences for the design of model translators. For illustration of general statements we use the structures of two kinematic model languages, NIRO and ROBOT. Set operations and predicates constitute the skeleton to describe, e.g. modifications of object structures or the functional dependencies of attributes.

Inhaltsverzeichnis

0. Einführung

Wir leben in einer Welt vielfältiger Abhängigkeiten. Ihre Beherrschbarkeit ist untrennbar verbunden mit realitätsnaher Information und korrektem Informationsaustausch. Das gilt in besonderer Weise, wenn der Computer bestimmte Funktionen des Menschen übernehmen soll. Eine computerintegrierte Fertigung (CIM) ohne angemessene Modellierung der Wirklichkeit ist undenkbar.
Offenbar ist es nicht einfach, eine komplette Fabrik und deren Produkt- und Fertigungsmittel zu beschreiben. Schon die statische Beschreibung als Momentaufnahme eines Weltausschnitts konfrontiert mit gigantischen Datenmengen und einer Fülle an komplexen Strukturen. Hinzu kommen eine Vielzahl von Abhängigkeiten zwischen den Daten, die schon allein durch eine Mehrfachbeschreibung auf verschiedenen Aggregationsstufen schwer zu vermeiden ist. Noch mehr Schwierigkeiten verursacht der Versuch, fehlerhafte Datenmanipulationen durch Zusatzbedingungen zu erkennen und angemessen zu behandeln.
Seit langem ist weltweit klar, daß der Wildwuchs immer neuer Systeme diese Probleme nicht lösen kann. Trotzdem kommen Standardisierungsbemühungen nur schrittweise voran. Ein Konsens ist bei der Fülle möglicher Wege so lange schwer zu finden, wie konsequent betriebene Abstraktion nicht zu den gemeinsamen Wurzeln der verschiedenen Ansätze vordringt. Sein Bedauern darüber kleidete Boos-Bavnbek auf der letzten GI-Jahrestagung in folgende Worte /Boos90/:
"Die computergestützte Modellierung ist so sichtbar verfahren, daß sich immer mehr Praktiker umfassende Analysen wünschen, die sich zu lehrbaren Qualitätskriterien und Handlungsweisen komprimieren lassen. Die Inadäquatheit derartiger Versuche liegt auch für den Praktiker auf der Hand. Es ist aber besser, sich an mächtige Ideen mit einer großen Macht innerhalb eines begrenzten Gültigkeitsbereichs zu halten, als ohne lehrbare Methodik auskommen zu müssen und auf Genialität angewiesen zu sein. Vor allem müssen durch methodisch bewußte Anwendung von erlernbaren Denkweisen einer gewissen Mächtigkeit Warnungen verbreitet und Schranken für die spontane, theorielose Gestaltung undurchschaubarer Systeme errichtet werden."
Auch wenn diese klaren Worte eines Mathematikers zum Widerspruch herausfordern, kann die Schlußfolgerung nur verstärkte Hinwendung zu gemeinsamen Grundstrukturen und integrationsfähigen Lösungsansätzen heißen. Daß das nicht von selbst geschieht, ist verständlich, muß doch dazu jeder Fachmann weit über seinen ursprünglichen Verantwortungs- und Tätigkeitsbereich hinausgehen, ohne davon immer sofort und direkt zu profitieren. Daß damit der Praktiker hoffnungslos überfordert wird, ist jedoch genauso falsch, denn in jedem Anwendungsfall spiegelt sich Allgemeines wider, daß durch eine passende Aufbereitung auch bewußt gemacht werden kann. Langfristig führt kein Weg an einem gemeinsamen Fundament vorbei, denn /Boos90/ "...in der Leichtigkeit von Produktion und der Schwierigkeit von Qualität liegt ein Potential für Chaos wie nie zuvor."
Die vorliegende Arbeit entstand nach mehrjähriger Auseinandersetzung mit realen Modelliersystemen und abstrakten Produktdatenmodellen. Sie stellt den Versuch einer persönlichen Bestandsaufnahme dar, die sich an der Maxime orientiert, daß "ein Ding dann gut ist, wenn nichts mehr wegzulassen ist" (Saint Exupery). Eine solche Einfachheit und Transparenz ist, wenn überhaupt, nur am Ende eines langen Prozesses erreichbar. Sie ist jedoch eine entscheidende Voraussetzung, ohne die eine Integration beliebiger Modelle schwer vorstellbar ist.

Naturgemäß gibt es eine Fülle von Beziehungen zu jedem beliebigen anderen Modellieransatz, deren gemeinsame Wurzeln die reale Welt mit ihrem einheitlichem Aufbau ist. Sie können hier nicht sehr vertieft werden, würde das doch auf die Schaffung eines allgemeingültigen Referenzmodells hinauslaufen, wie es etwa für CAD-Systeme in einer ersten Ausbaustufe bereits existiert /Refe89/ und wie es in Gestalt einer offenen Systemarchitektur etwa in CIM- OSA angestrebt wird. Schwerpunkte der Arbeit sind nach einer kurzen Erläuterung der Problemlage

- ein objektorientiertes Konzept zur Beschreibung integrationsfähiger Datenmodelle unter Berücksichtigung möglicher Zwangsbedingungen
- eine grafische Modelliersprache zur anschaulichen Beschreibung von Welt- und Modellstrukturen
- eine Analyse objektorientierter Sprachstrukturen und ihres Zusammenhangs mit Grundstrukturen zugehöriger Übersetzer
- eine Veranschaulichung der allgemeinen Aussagen am Beispiel kinematischer Strukturen, die als Netzstrukturen transparenten Aufbaus dafür besonders geeignet sind.

Dazu werden im ersten Kapitel zunächst einige Anforderungen zusammengestellt, die sich bei einer integrierten Verarbeitung von Produktdatenmodellen ergeben.
Es folgt im zweiten Kapitel eine Erörterung der Grundstrukturen von Produkten und Datenmodellen und deren wechselseitigen Zusammenhangs.
Das dritte Kapitel dient der Vorstellung wesentlicher Elemente einer grafischen Beschreibungssprache ("pUR-Modellsprache"), die trotz ihres einfachen Aufbaus eine große Mächtigkeit und hohe Flexibilität aufweist.
Im vierten Kapitel werden einige Konsequenzen diskutiert, die sich aus der klaren Strukturierung von Produktdatenmodellen für den Bau von Modellübersetzern ergeben.
Das fünfte Kapitel schließlich dient der Veranschaulichung der allgemeinen Aussagen, für die als Beispiel zwei Kinematikmodell- Strukturen NIRO und ROBOT zugrundegelegt wurden.
Die schwerpunktmäßige Orientierung auf die Modelldefinitionsebene bedeutet nicht, daß die Grundlegung ohne Berücksichtigung der Anforderungen erfolgt, die von der Seite der Modellmanipulation und der Modellsimulation her kommen. Mengenoperationen und Prädikate bieten ein Grundgerüst, um etwa Strukturänderungen der Objekte oder funktionelle Abhängigkeiten von Attributen zu beschreiben. Eine stärkere Vertiefung ist im vorliegenden Rahmen jedoch nicht möglich.
Für den Klärungsprozeß bot ein einjähriger Aufenthalt am Kernforschungszentrum Karlsruhe (KfK) optimale Voraussetzungen. Er gab mir Gelegenheit, mich eingehender mit dem neuen STEP- Kinematikmodell und der Compilerbau-Toolbox der Forschungsstelle Karlsruhe der GMD zu beschäftigen /GrEm90/. Am Institut für Reaktorentwicklung IRE (Leitung : Prof. Smidt) fand ich insbesondere in Herrn Dr. Schlechtendahl und Herrn Gengenbach aufgeschlossene Partner, die durch Anregungen und Widerspruch wesentlich zur Klärung bisheriger Vorstellungen beitrugen.

1. Über Produktdatenmodelle für CIM-Lösungen

1.1. Die Bedeutung von Produktdatenmodellen

Eine Beschreibung technischer Produkte durch geeignete Datenmodelle ist kein neues Problem. Wichtige Teilergebnisse liegen seit langem vor, wurden ja bereits bei der Nutzung von CA*-Techniken geeignete Partialmodelle benötigt. Deren großer Vorteil war, daß sie wegen ihrer Nähe zur konkreten Anwendung effizient und ohne überflüssigen Datenballast gestaltet werden konnten.
Relativ früh wurde erkannt, daß solche Insellösungen nur begrenzte Effekte zulassen. Als entscheidender Mangel aus Unternehmenssicht erwies sich, daß Partialmodelldaten nicht durchgängig von verschiedenen Systemen aus genutzt werden können. Ohne wesentliche Effizienzverluste im ursprünglichen Nutzungsbereich müssen also Daten unterschiedlichster Anwender so zusammengeführt werden, daß eine multivalente Nutzung möglich wird. Damit entstehen eine Reihe grundsätzlicher Fragen nach der Auswahl und Strukturierung von Produktdaten, deren Beantwortung ohne stärkere Abstraktion nicht möglich ist.
Die Schaffung kompletter Produktbeschreibungen innerhalb eines Modells ist eine Aufgabe der Zukunft. Sie ist nur in Etappen zu erreichen, wobei Skeptiker ein solches Ziel immer wieder in Frage stellen. Favorisiert werden häufig Vorgehensweisen, die unterschiedlichste Modellbildungen nebeneinander zulassen wollen. Auf diese Weise sollen unnötige Einschränkungen bei der Schaffung neuer Modelle und eine Entwertung vorhandener, bewährter Systeme vermieden werden.
In der Praxis gibt es seit 1984 im Rahmen der ISO (International Standardisation Organisation) seitens des Komitees ISO/TC184/SC4 große Anstrengungen, den Produktdatenaustausch durch Bereitstellung eines Standards STEP (Standard for the Exchange of Product Model Data) international zu regeln /AnSc87/. Vorausgegangen war eine Schaffung nationaler Schnittstellenstandards für wichtige Partialmodelle, etwa im Bereich der Geometrie (IGES, VDAFS, SET) oder produktionstechnischer Daten (CLDATA,IRDATA), die eine weite Verbreitung gefunden haben, ohne das Gesamtproblem zu lösen. Wichtige Anregungen gingen darüber vom ESPRIT- PROJEKT "CAD-Interfaces" aus, das eine verstärkte Einflußnahme auf den internationalen Standardisierungsprozeß ermöglichte /Schl88/. Es ist legitim, durch Schaffung von Schnittstellen und Bereitstellung geeigneter Übersetzer eine Verkoppelung von Systemen zu unterstützen /Scho88/. Bei der Schaffung neuer Systeme sollten jedoch ganz bewußt Datenmodelle so strukturiert werden, daß ihre Integration mit einem Minimum an Aufwand möglich wird. Ohne ein sauberes begriffliches und methodisches Fundament ist diese Aufgabe nicht zu lösen.

1.2. Zur Systemverknüpfung über Produktdatenmodelle

Wie sieht nun die Verknüpfung verschiedener Systeme bei der Lösung übergreifender Aufgaben aus? In reiner Form ist eine Kopplung von Systemen entweder auf der Ebene der Kommunikation oder der Ebene der Kooperation möglich. Im ersten Fall erfolgt eine direkte Weitergabe der Modelldaten an andere Systeme, was auf die Bildung und den Austausch kompletter Modellkopien hinausläuft. Offenbar ist dieser Weg recht gut geeignet, um unter Erhaltung des status quo bereits existierende Systeme zu verknüpfen. Dafür

sind geeignete Schnittstellen für den Datenaustausch festzulegen, auf deren Basis Datenfolgen vorgeschriebener Struktur korrekt umgewandelt werden können.
Im zweiten Fall ist ein einheitlicher Zugang zu einem Produktdatenmodell von verschiedenen Systemen aus zu sichern. Das bedeutet bei voller Ausbaustufe eine Datenbanklösung für ein integriertes Produktdatenmodell mit verteilter Verarbeitung. Das Vorhandensein eines adressierbaren Datenspeichers schafft dabei gegenüber sequentiellen Verarbeitungstechniken sehr viel effizientere Möglichkeiten des Datenzugriffs. Dies ist für viele Ansendungen entscheidend, ohne daß jedoch of logischer Ebene prinzipielle Unterschiede bestehen. Eine Struktur des Datenmodells kann daher, notfalls unter Verwendung von Rückverweisen im Falle von Netzstrukturen, als lineare Sequenz ausgegeben werden, so daß bei Wiedereinlesen eine Rekonstruktion der Struktur möglich wird.
Das Herauslösen von Teilstrukturen als Mischform beider Fälle besitzt eigenständige Bedeutung. Auf diese Weise können auf Zeit Kopien einem Anwendungssystem zur Verfügung gestellt werden, um am Ende verändert in das Produktdatenmodell zurückzugelangen (CHECK IN, CHECK OUT). Eine wichtige Rolle spielen insbesondere Partialmodelle mit einer relativen Autonomie, die getrennte Sichten auf ein Produkt realisieren. Darüberhinaus sind in bestimmten Fällen Beschreibungen solcher Modellauszüge auch durch Programme sinnvoll, wenn etwa aufgrund hoher Abhängigkeiten aus wenigen Basisdaten viele andere Daten berechnet werden können.
Ein zentrales Problem ist in allen Fällen die Wandlung der Beschreibungsdaten, um den Konventionen verschiedener Anwendersysteme zu entsprechen. Dazu ist zunächst eine Überführung der inneren Modellrepräsentation in eine externe nötig. Die externen Beschreibungsdaten müssen die Bedingungen einer Sprache erfüllen mit eindeutiger Syntax und klarer Semantik. Die Anpassung an ein konkretes Anwendersystem erfordert dann Übersetzer, die die verschiedenen Produktbeschreibungen ineinander überführen. Die Datensequenz einer Quellsprache muß also in eine Datensequenz der Zielsprache umgewandelt werden, die die ursprüngliche Bedeutung der Daten richtig widerspiegelt. Es ist naheliegend, solche Übersetzer schnell und möglichst rechnergestützt aus einer Schnittstellenspezifikation generieren zu lassen. Leistungsfähige Werkzeuge stehen inzwischen zur Verfügung und werden mittlerweile einer breiten Öffentlichkeit offeriert /Ge1191/, doch erfordert ihre Nutzung nach wie vor ein tiefes Verständnis für die ablaufenden Prozesse. Insbesondere kann dem Nutzer auch auf diesem Niveau nicht die Bereitstellung von Programmcode erspart werden.

1.3. Zur Konsistenzsicherung bei Produktdatenmodellen

Von besonderer Brisanz ist die Frage der Konsistenzsicherung. Es muß gewährleistet werden, daß,

- die Modelldaten ein formal korrektes Modell des zu beschreibenden Produkts bilden
- physikalische Gesetzmäßigkeiten, funktionsbezogene Zwangsbedingungen und anwendungsgerechte Randbedingungen nicht verletzt werden.
- verschiedene Partialmodelle eines Produkts untereinander verträglich bleiben.

Für diesen Zweck macht sich die Aufnahme zusätzlicher Informationen notwendig, um im Änderungsfall Widersprüche feststellen zu können. In einer höheren Ausbaustufe wäre dann deren direkte Behebung ohne Anwendereingriff wünschenswert.

Eine gewisse Entlastung bietet eine Einführung standardisierter Partialmodelle, deren Abgleich im Rahmen einer speziellen Verträglichkeitsüberprüfung vorgenommen werden kann. Dieser Weg wird augenblicklich bei der Realisierung von STEP beschritten. Eine Menge unabhängiger Beschreibungsdaten wird so auf verschiedene Partialmodelle verteilt, daß der Attributedurchschnitt je zweier Partialmodelle leer bleibt. Durch wechselseitige Referenzierbarkeit ist es möglich, Beschreibungsdaten eines Partialmodells mit denen eines anderen zu assoziieren. Bei Auflösung der Referenz kann also auf Daten eines anderen Partialmodells zurückgegriffen werden, die nun gemeinsam mit eigenen Daten im aufrufenden Modell Bedingungen unterworfen werden können.
Bestimmte Basismodelle können in STEP von verschiedenen Anwendungsmodellen übernommen werden, die eigene Daten hinzufügen. Das bedeutet, bildlich gesprochen, nichts anderes, als daß der Nutzer sich einer allgemeinen Umgangsspache bedient, für die Erweiterungen um seinen Fachjargon existieren. Notwendige Änderungen an den Daten eines Basismodells innerhalb auch nur eines Anwendungsmodells machen eine Wiederholung der übrigen Anwendungsfälle notwendig, was mittels eines Lebenszyklusmodells abgesichert werden kann.
Hauptproblem einer Konsistenzsicherung unter diesen Annahmen ist, daß die Beschreibung irgendeines realen Produktteiles in verschiedenen Partialmodellen gleichzeitig erfolgt. Da ein Zusammenhang der Attribute mit konkreten Objekten nicht explizit festgehalten wird, ist etwa im Falle des Löschens eines Produktteils nur bei vorbereiteter Querreferenzierung durch alle Modelle hindurch erkennbar, welche Attribute getilgt werden müssen. Weiter ist zu berücksichtigen, daß die als unabhängig deklarierten Attribute häufig Zusatzforderungen unterliegen, die modellübergreifend sind. Bei Verwendung besseren Materials etwa kann ein Produktteil kleiner dimensioniert werden, so daß die Kosten insgesamt geringer werden.
Bedingungen zur Konsistenzsicherung haben noch aus einem anderen Grund Bedeutung. Die Zwangsbedingungen können so verschärft werden, daß klare funktionelle Abhängigkeiten zwischen Attributen entstehen. Sind also alle Eingangsattribute einer Funktion bekannt, ergeben sich zwangläufig und eindeutig die zugehörigen Ergebnisattribute. Damit ist bereits eine Brücke zu einer Modellwelt mit veränderlichen Zuständen geschlagen. Es lassen sich jetzt dynamische Modelle aufbauen, die durch Anweisungsfolgen einer Programmiersprache, mittels Petrinetzen, mit den Mitteln einer objektorientierten Programmierung oder auf andere Weise realisiert werden können. Insbesondere ist es möglich, alle Konstrukte einer höheren Programmiersprache auf der Grundlage derartiger Spezifikationen abzuleiten /DiFe88/.

2. Grundstrukturen eines Integrierten Produktdatenmodells

2.1. Zum Begriff des Produktdatenmodells

Der Begriff "Produktdatenmodell" erscheint zunächst einfach interpretierbar. Offenbar ist ein Modell eines Produkts gemeint, das aus einer Menge produktbeschreibender Daten besteht. Es ist vorstellbar, daß diese als Zeichenfolgen einer Produktbeschreibungssprache, in Form einer grafischen Darstellung oder als Dateneinträge eines Speichers vorliegen. Das Modell beschreibt dann im CIM-Umfeld ausgewählte Anforderungen an ein reales oder realisierbares Produkt. Dafür ist eine Transformation

TR: MODELL ⟶ PRODUKT

notwendig, die die Abhängigkeit bestimmter Objektattribute von den Modelldaten festhält. Umgekehrt legt jedes konkrete Produkt bestimmte Eigenschaften der ihm zugeordneten Modelle fest. Die praktische Nutzung erfordert dafür eine Rücktransformation

RT: PRODUKT ⟶ MODELL.

Der Zusammenhang von Produkt und Modell läßt sich dann durch ein einfaches Schema darstellen (Bild 1).

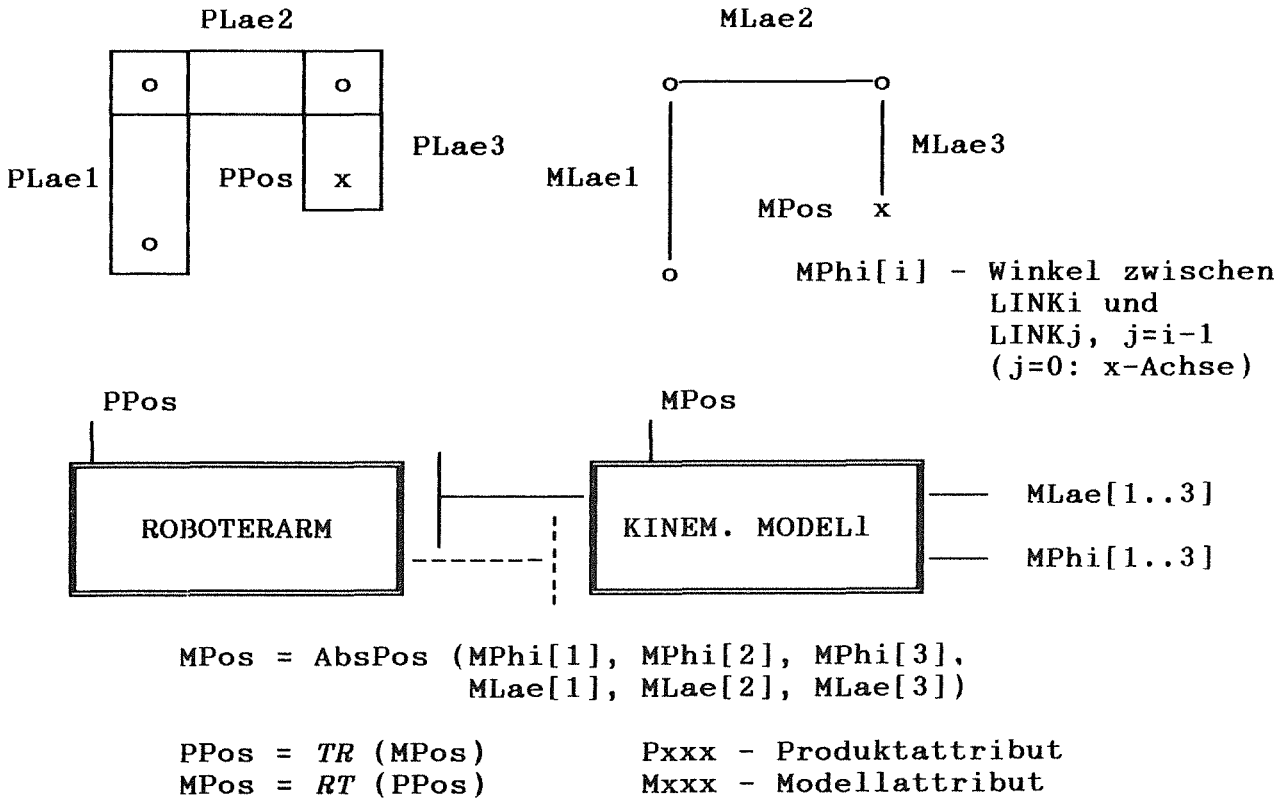
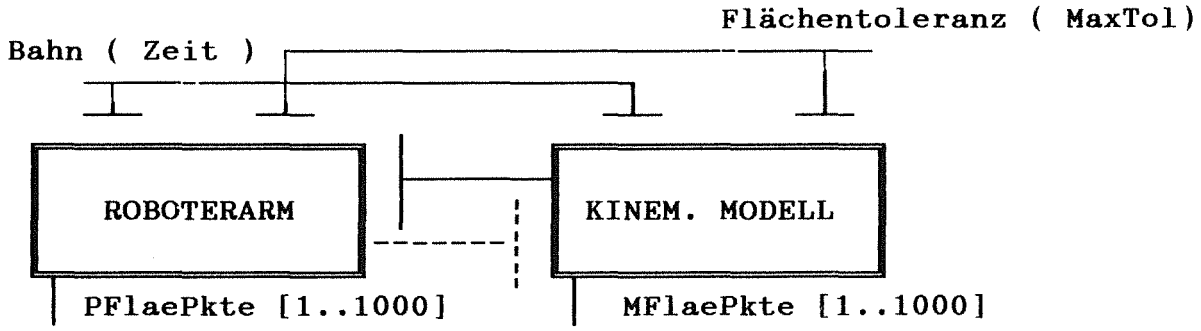


Bild 1.Zusammenhang von Produkt- und Modellattributen

Durch die Notation soll anschaulich zum Ausdruck gebracht werden, daß bestimmte Attribute des Produkts nicht frei gewählt werden können, sondern von Vorgaben des Modells abhängen (und umgekehrt). So reicht für kinematische Untersuchungen ein Modell aus, in dem relative Lage und Freiheitsgrade der Koppelstellen ("Pairs") der beweglichen Glieder ("Links") festgehalten sind. Mit seiner Hilfe läßt sich unmittelbar im Modell ableiten, wie etwa die Werkzeugposition MPos zu einem definierten Zeitpunkt sein sollte. Damit kann nicht nur aus konkret gespeicherten Modellattributen wie MLae[1] oder MPhi[3] auf die reale Länge PLae[1] und den realen Winkel PPhi[3] des Roboterarms geschlossen werden. Es ist jetzt auch in erster Näherung möglich, einer berechneten Größe wie MPos bei bekannter Transformationsvorschrift *TR* eine zunächst unbekannte Produkteigenschaft PPos zuzuordnen, ohne daß deren Messung am konkreten Produkt notwendig wird.
Allgemein ist ein solcher Zusammenhang in einem Metamodell darstellbar, das beliebige, auch nichtfunktionelle Beziehungen, zwischen Modell- und Produktattributen zu beschreiben gestattet. So wird ein Roboterarm für jeden Punkt der Oberfläche eine Abweichung Abst vom Sollwert haben, deren maximale Größe durch eine Toleranzvorgabe MaxTol festgelegbar ist (Bild 2). Die Vorgabe führt auf eine Einschränkung für die mögliche Lage beliebiger Produktpunkte, die (unter Berücksichtigung der Abbildungsvorschrift *RT*) in Beziehung zu bekannten Modellpunkten gebracht werden.



$$| \text{Abst}(RT(\text{PFlaePkte}[i]), \text{MFlaePkte}[i]) | < \text{MaxTol} \qquad i = 1, \ldots 1000$$

$$RT(\text{PPos}(\text{Zeit})) - \text{AbsPos}(\text{Zeit}, \text{MPhi}[1], \ldots \text{MLae}[3]) = 0$$

Bild 2. Vorgabe von Zwangsbedingungen in nichtfunktioneller Form

Ein Grundproblem jeder Modellierung besteht darin, daß zwischen konkreten Produkten und konkreten Modellen vielfältige Beziehungen existieren. Sie werden im allgemeinen zweckgebunden hergestellt, um Aussagen über Struktur und Verhalten eines Produkts ableiten zu können, die über einen direkten Produktrückgriff nicht, noch nicht, nicht mehr oder nur schwer zu beschaffen sind. So werden kinematische Untersuchungen zu einem Zeitpunkt durchgeführt, wo das Produkt noch nicht existiert. Wesentlich dabei ist die Tatsache, daß die Attribute eines Modells und eines Produkts völlig verschiedener Natur sein können. Das bietet insbesondere im Falle eines Datenmodells die Möglichkeit, unter-

schiedliche physikalische Eigenschaften wie Zeitpunkt, Lage oder Farbe durch gleichwertige Speicherelemente zu erfassen.
Offenbar wird durch ein Modell nicht nur ein Einzelprodukt festgelegt, sondern eine ganze Klasse von Produkten. Das hängt damit zusammen, daß eine im Modell nicht spezifizierte Eigenschaft beliebig geändert werden kann. Dasselbe gilt, wenn eine Eigenschaft zwar festgehalten wird, jedoch innerhalb vorgegebener Grenzen variieren kann. Fehlt etwa eine Lagevorgabe, beschreibt das Datenmodell eines Musterteils gleichzeitig sämtliche reale Kopien dieses Teils. Sind nur die maximalen Ausdehnungen eines Körpers festgelegt, kann seine Form noch auf vielfältige Weise präzisiert werden (Bild 3).

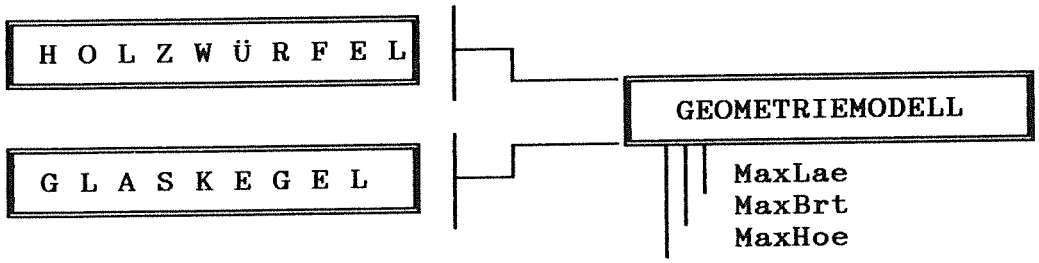


Bild 3. Verschiedene Realisierungen zu einem Modell

Umgekehrt kann auch ein konkretes Produkt durch mehrere Modelle festgelegt werden. Das ist insofern der Normalfall, als jedes Modell nur bestimmte Eigenschaften eines Produkts widerspiegelt. So entstehen voneinander getrennte Partialmodelle mit unterschiedlicher Attributeauswahl, die den Anforderungen bestimmter Anwendergruppen genügen (Bild 4).

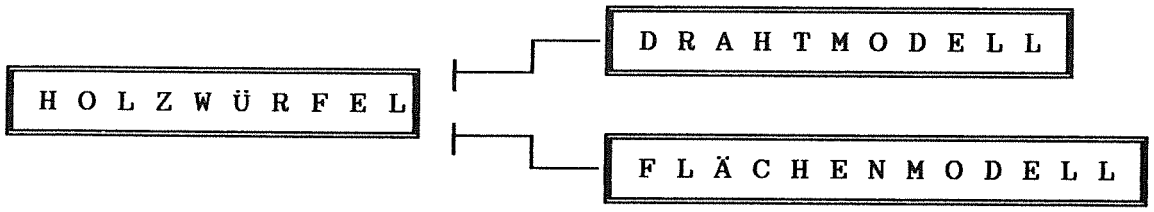


Bild 4. Verschiedene Geometriemodelle für ein Produkt

Als Grenzfall können aber auch austauschfähige Partialmodelle mit gleichem Informationsgehalt festgelegt werden, deren Attribute dann wechselseitig voneinander abhängen. Sie erlauben eine Produktbeschreibung, die auf die Anforderungen der jeweiligen Umgebung besonders zugeschnitten ist. So lassen sich Kurven im allgemeinen besser als große Zahlentabellen durch den Menschen auswerten, und ein Amerikaner muß nicht notwendig einen deutschen Sachtext verstehen (Bild 5). Voraussetzung für eine derartige Wandlung ist das Vorhandensein geeigneter "Übersetzer", auf die an späterer Stelle näher eingegangen werden soll.

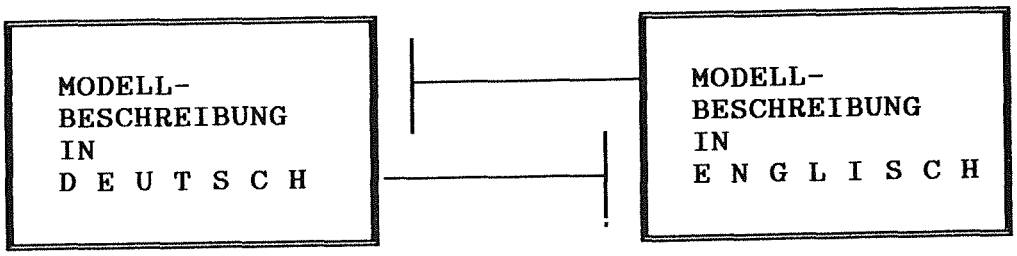


Bild 5. Äquivalente Modelle (aus Sicht einer Sachinformation)

Besondere Aufmerksamkeit verlangt die Tatsache, daß ein Produkt unterschiedliche Zustände annehmen kann. Das Modell sollte durch Einführung von Zwangsbedingungen möglichst sichern, daß Verletzungen dieser Forderungen kenntlich gemacht werden können. So kann ein Drehgelenk nicht beliebig weit geöffnet werden, ohne daß die gekoppelten Arme kollidieren. Damit ergibt sich eine Beziehung zwischen realen Objekteigenschaften, die auf das Modell übertragen werden muß (Bild 6a). Daneben lassen sich durch parameterabhängige Bedingungen beliebige "künstliche" Anforderungen formulieren (Bild 6b). Insbesondere ist es mit Hilfe solcher Anforderungen möglich, aus einer Auswahlmenge möglicher Objekte mit relativ wenigen Informationen einen passenden Vertreter zu bestimmen.

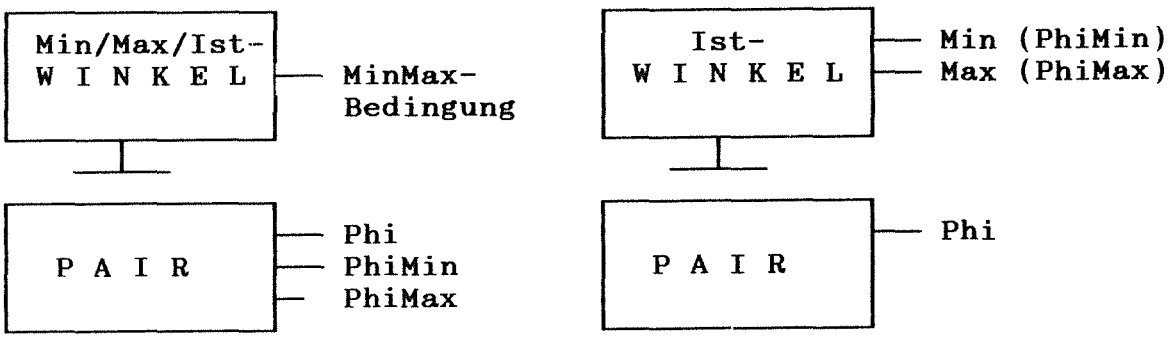


Bild 6. Zwangsbedingung a) ohne Attribute b) attributbehaftet

2.2. Modellklassen und Modellinstanzen

Für die praktische Arbeit, z.B. eine Abspeicherung oder Verarbeitung der Datenmodelle ähnlicher Produkte, haben sich Modellschemata bewährt. Sie legen den Aufbau von Modelleinheiten (Entities) fest, indem zulässige Attributetypen und referenzierbare Entitytypen angegeben und zusätzlich mögliche Einschränkungen erfaßt werden. Bei STEP etwa geschieht das unter Verwendung einer eigenen Modelliersprache EXPRESS, die insbesondere über Deklarationen und Regeln geeignete Festlegungen zuläßt. Auf dieser Grundlage können unterschiedliche Attributewerte oder sogar Teilstrukturen

konkreter Produktdatenmodelle so festgelegt werden, daß eine Überprüfbarkeit möglich wird (Bild 7).

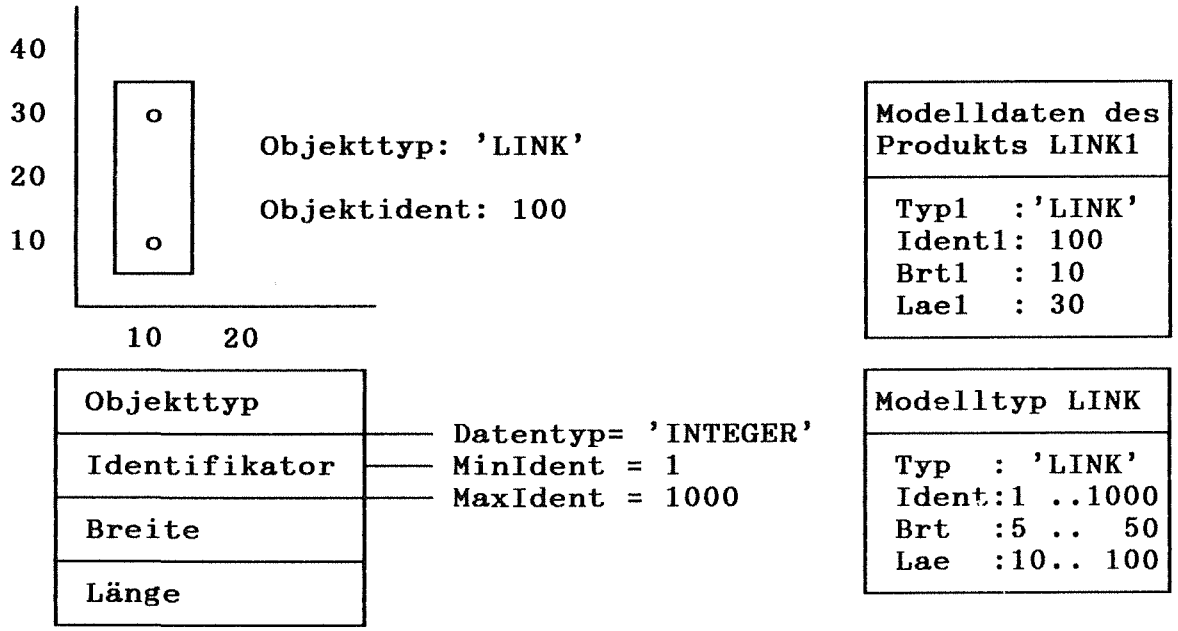


Bild 7. Vereinbarung eines Modellschemas

Die Einführung des Modellschemas führt dazu, daß gelegentlich der Begriff "Produktdatenmodell" als Synonym für "Modellklasse" aufgefaßt wird. Damit wird automatisch eine Mengenvorstellung assoziiert, der dann notwendig "Modellinstanz" als Einzelexemplar gegenübergestellt werden muß. Daraus können sich Mißverständnisse ergeben, da bei einer konkreten Modellierungsaufgabe tatsächlich nach einer passenden Modellinstanz gesucht wird.
Die Schwierigkeiten entstehen durch zwei unterschiedliche Sichten, die bei der Beschreibung der Realität möglich sind. Zum einen kann von einem Einzelprodukt ausgegangen werden, für das alle Eigenschaften im benötigten Umfang gesammelt werden sollen. Das Produktdatenmodell ist dann Abbild eines Einzelprodukts, das durch unterschiedlich präzise Attributevorgaben unterschiedlich stark festgelegt sein kann.
So können statt eines konkreten Attributs über ein Prädikat Anforderungen an das Attribut festgelegt werden. Dadurch können die Ausprägungen eines Exemplars differieren, ohne daß die Anforderungen verletzt werden. Die zulässigen Grenzen werden durch einen Modelltyp beschreibbar, der selbst noch kein reales Produktdatenmodell ist, sondern erst durch Konkretisierung seiner Attribute und Teilstrukturen dazu wird.
Zum anderen kann von einem Einzelattribut ausgegangen werden, für das alle Produkte mit dieser Eigenschaft zusammengestellt werden. Es wird damit insbesondere eine Auswahlmenge beschreibbar, wie sie typischerweise bei der Realisierung von Datenbanken auftritt. Statt eines konkreten Attributs kann wieder von einer Menge von Attributen ausgegangen werden, die durch eine Prädikatevorgabe zugelassen wird. Klasse heißt dann nichts anderes als Klasse von möglichen Exemplaren, aus der heraus ein Vertreter auszuwählen ist. Werden die Bedingungen verschärft, entstehen Subklassen, werden sie gelokkert, ergeben sich Superklassen. Zu jedem Modelltyp existiert im allgemeinen eine Klasse von Modellen, deren Exemplare den Anforderungen des Typs genügen.

Ein Produktdatenmodell soll im weiteren immer Repräsentant eines konkreten Produkts sein, zur Betonung diese Sachverhalts werden wir jedoch auch manchmal von Modellexemplar, Modellinstanz oder Ausprägung eines Modelltyps sprechen.

2.3. Zur Integration von Produktdatenmodellen

Im einzelnen soll ein Produktdatenmodell und damit sein Schema sowohl die anwendungsspezifische Teilestruktur des Produkts als auch ausgewählte Eigenschaften und Beziehungen der Produktteile widerspiegeln. Insbesondere sollen auf höherer Abstraktionsstufe mögliche Alternativen für ausgewählte Produktteile zugelassen sein. In idealisierter Form enthält es alle relevanten Beschreibungsdaten des Produkts, deren Konsistenzsicherung und Verfügbarmachung für unterschiedliche Nutzergruppen seine Hauptschwierigkeit ausmacht.
Die Möglichkeit, für ein beliebiges Produkt ein Produktdatenmodell maßgeschneidert festlegen zu können, wird den Anforderungen von CIM noch nicht gerecht. Es bleibt die entscheidende Frage, wie die Zusammenhänge der unterschiedlichen Modelle beherrscht werden sollen. Dafür ist der Übergang auf eine Meta-Ebene notwendig, die die Wechselbeziehung zwischen verschiedenen Partialmodellen zu beschreiben gestattet.
Offenbar ist der Aufwand groß, mehrere Partialmodelle eines Produkts untereinander verträglich zu halten. Werden je zwei Modelle miteinander abgestimmt, kann das bei n Modellen bis zu $n*(n-1)/2$ Paarungen bedeuten, die im Änderungsfall betroffen sein können. Dabei sind Iterationen und Adaptionen zur Widerspruchsbehandlung noch nicht berücksichtigt. Unter diesen Umständen ist es wünschenswert, ein gemeinsames Bezugsmodell für Änderungen zu nutzen. Ein solches "Integriertes Produktdatenmodell" sollte die Änderungen von Partialmodellen übernehmen können und die Verträglichkeit mit allen anderen Partialmodellen sichern.
Bei der Festlegung verschiedener Partialmodelle sind zwei Grenzfälle denkbar, die unmittelbar Einfluß auf dem Datenumfang eines Integrierten Produktdatenmodells haben. Zum einen können die verschiedenen Partialmodelle dieselben Eigenschaften eines Produkts charakterisieren, jedoch in unterschiedlichen Systemen und in unterschiedlicher Form. So kann eine Gerade durch zwei Punkte oder einen Punkt, Länge und Winkel festgelegt werden. Zum anderen können die verschiedenen Partialmodelle unabhängige Eigenschaften enthalten, die nicht auseinander ableitbar sind. Offenbar ist nur im zweiten Fall eine Übernahme aller Partialmodellattribute in das Integrierte Modell notwendig. Im ersten Fall wird nur die Hälfte der Beschreibungsdaten benötigt, dafür ist jedoch eine Datenwandlung notwendig.
Die Anforderungen an ein solches Integriertes Modell sind naturgemäß hoch /Grab90, Ande89/. Das Modell müßte mindestens alle unabhängigen Beschreibungsattribute des Produkts in geeigneter Strukturierung enthalten. Es sollte möglichst auch nicht mehr enthalten, da abhängige Attribute ableitbar sind und in Partialmodellen untergebracht werden können. Überlappende Produktteile, wie sie etwa schon bei Aggregation als Sonderfall einer Vernetzung entstehen, führen jedoch zwangsläufig zu Abhängigkeiten. Sie können gezielt dazu genutzt werden, um Attribute von Einzelteilen wegzulassen und durch Attribute zu ersetzen, die Relationen zwischen mehreren Teilen charakterisieren. Werden die so ersetzten Attribute in einem Partialmodell benötigt, können sie bei Kenntnis der Zusammenhänge dort abgeleitet werden.
Die Wertebelegung der Attribute eines Integrierten Modells erfordert Zugriffsfunktionen, die auf die Attribute von Partialmodellen zurückgreifen. Umgekehrt wäre im Änderungsfall zu sichern, daß die Attribute aller betroffenen Partialmodelle nachgeführt werden

müssen. In jedem Fall hätte man es bei n Partialmodellen nur noch mit maximal n Zugangspaarungen und n Abgangspaarungen von Modellen zu tun. Dafür wäre jedoch ein zusätzliches Modell aufzubauen, das zu einer Doppelung der Attributespeicherung führt.
Die hier angesprochenen Probleme beschäftigen bereits seit langem auch STEP- Entwickler /GrAS89/. In der Vergangenheit wurde ein Integriertes Produkt-Informationsmodell IPIM konzipiert, das zwar die vollständigste bekannte und formalisierte Beschreibung von Weltausschnitten realisiert, jedoch das Problem der Zusammenführung unterschiedlicher, in getrennten Schemata beschriebener Sichten auf ein Produkt nicht befriedigend löste. Es fehlte die Möglichkeit einer Generalisierung, durch die etwa jede Sicht als Spezialisierung eines gemeinsamen 'generic object' kenntlich gemacht werden konnte. Insbesondere war es in einem zentralen IPIM-Teilschema PSCM (Product Structure Configuration Management) auf Grund fehlender sprachlicher Mittel nicht möglich, Versionen, Varianten oder Alternativen von Teilebeschreibungen zu unterscheiden /Blen90/.
Die aktuellen Bemühungen um eine Integration von STEP- Partialmodellen (Resource Models, Application Models) konzentrieren sich zur Zeit auf GPDM (Generic Product Data Model) /DaYa90/ und seine Derivate wie GPDR /DaYa91/. Dabei werden Zusammenhänge zwischen den Modellen durch Schemaerweiterungen ausgedrückt. Aus Datenbanksicht ist seit längerem klar, daß die klassischen Datenmodelle (Hierarchisches Modell, Relationenmodell, Netzwerkmodell) nur eine begrenzte Ausdrucksfähigkeit bieten. Für Produktdatenmodelle werden semantische Modelle benötigt, die

- den statischen Aufbau ebenso wie
- die dynamische Veränderung realer Weltausschnitte auf hohem Niveau widerspiegeln und
- durch Integration aller Aspekte eine einheitliche Handhabung ermöglichen /KiMc85/.

2.4. Grundstrukturen eines Integrierten Produktdatenmodell

Die Vereinigung der Daten unterschiedlicher Partialmodelle in einem Modell führt auf das Problem, welche Daten konkret zusammengefaßt werden sollen. Für die Lösung bietet sich ein objektorientierter Ansatz an. Dazu wird von irgendeiner Zerlegung des Produkts in Teile ("Objekte") ausgegangen, die der Menge von Modelleinheiten ("Entities") gegenübergestellt wird. Für jedes Entity ist das i.a. kleinstmögliche Objekt zu bestimmen, das den im Entity beschriebenen Eigenschaften zugrundeliegt (Bild 8). Damit können unterschiedliche Partialmodelle auf dem Umweg über gemeinsame Objekte in Wechselbeziehung gebracht werden.
Das Integrierte Modell würde sich daraus ergeben, indem neue Entities ("Units") festgelegt werden, die genau diese Objekte repräsentieren. Jede solche Unit übernimmt die Attribute sämtlicher Partialmodellentities, die auf das zugehörige Objekt Bezug nehmen. Die Attribute sollten einerseits möglichst unabhängig sein, andererseits alle Attribute der Partialmodelle verarbeiten können.

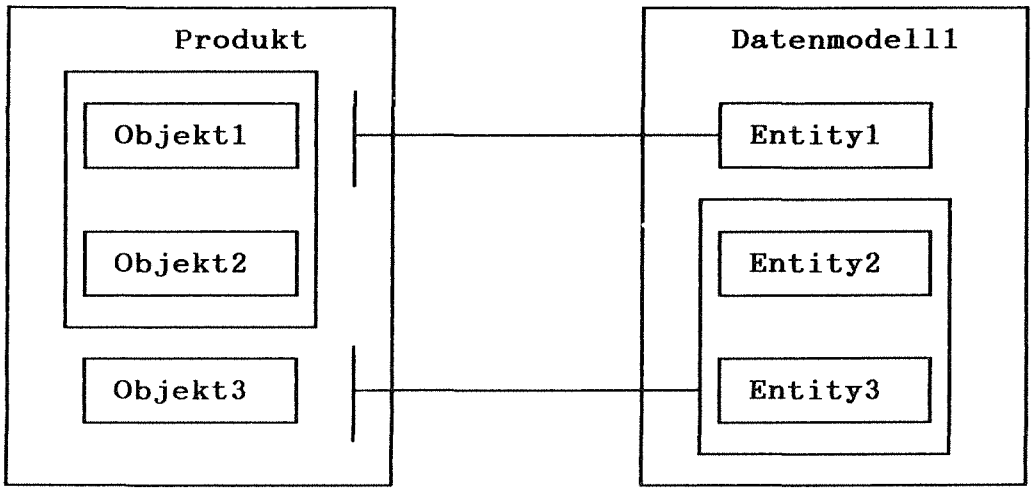


Bild 8. Zuordnung von Modellentities zu Produktobjekten

Die zentrale Bedeutung eines Integrierten Modells legt eine möglichst übersichtliche Objektzerlegung des Produkts nahe. Eine solche läßt sich durch mehrstufiges Zerschneiden, d.h. eine orthogonale, rekursive Zerlegung jedes Objekts, erreichen. Sie liefert für die zugehörigen Units eine Baumstruktur, bei der beliebige Units (bis auf die Wurzelunit) genau Teil einer Unit der nächsthöheren Stufe sind. Andere Units, die durch Rückgriff auf Teile bereits bestehender Units möglich sind, werden damit ausgeschlossen. Dies bedeutet im Falle einer sprachlichen Beschreibung, daß eine Kennzeichnung durch Begriffe nur für durchschnittsfremde Objekte einer Aggregationsstufe existiert, nicht jedoch für Objekte, die aus Teilen unterschiedlicher Objekte dieser Aggregationsstufe zusammengesetzt sind. Es ist leicht vorstellbar, daß auf diese Weise der Zusammenhang der Attribute verschiedener Objekte wesentlich einfacher zu durchschauen ist. Allerdings wird der Aufwand größer, Attribute aus Partialmodellen zu übernehmen, die hier nichtorthogonalen Objekten zugeordnet sind. Das mag mit ein Grund dafür sein, daß hierarchische Datenbanken als ungeeignet im CIM- Umfeld gelten /Scho89/. Trotzdem erscheint es reizvoll, die Vor- und Nachteile orthogonaler Modellstrukturen für integrierte Lösungen noch einmal sorgfältig miteinander zu vergleichen.
Die Beschreibung der Modellbildung erfordert einheitliche Darstellungsmittel für Objekte und Entities. Sie lassen sich relativ zwanglos einführen, wenn auch Modelle als konkrete Produkte aufgefaßt werden. Dazu werden zunächst getrennt für Produkt und Datenmodelle pUR-Modelle erzeugt, die trotz unterschiedlicher Realität der Objekte auf eine einheitliche Beschreibungsform führen. In einem "Hypermodell" lassen sich damit Entities gleichfalls als Units ("Modellunits") ansehen, die den Units des Integrierten Produktdatenmodells ("Produktunits") an die Seite gestellt werden können. Durch Auflösung der Referenzen ("Reduzierung") lassen sich Modellunits als zusammenhängender Bereich vorstellen, in dem alle Attribute der jeweiligen Modellunit gemeinschaftlich bereitstehen.
Die nächste Abstraktionsstufe ergibt sich daraus, daß ein Zusammenhang zwischen Produktunits und Modellunits hergestellt werden muß. Dazu sind, wieder auf der Ebene von pUR- Modellen, Äquivalenzen anzugeben, die Units und deren Eigenschaften wechselseitig aufeinander abzubilden gestatten. Setzen wir voraus, daß die Produktunits Struktur und Eigenschaften des Produkts hinreichend genau widerspiegeln und damit die Voraussetzungen eines Integrierten Produktdatenmodells erfüllen, bedeutet dies nichts anderes als die Schaffung von Metamodellen für die Partialmodelle. Mit ihrer Hilfe kann jetzt die Sem-

antik der Partialmodelle beschrieben werden, deren gemeinsame Grundlage ja das zu modellierende Produkt bildet. Die Beziehungen zwischen den einzelnen Partialmodellen können schließlich im Rahmen eines allgemeinen Metamodells des pUR-Hypermodells festgelegt werden, worin der Zusammenhang mit dem Integrierten Produktmodell eingeschlossen ist. (Bild 9).
Jede Unit steht zunächst für eine eigenständige Modelleinheit. Ist das anders auf Grund eines existentiellen Zusammenhangs der zugeordneten Objekte , muß das durch Angabe von Unitbeziehungen ("Relationships") festgehalten werden. Auf diese Weise kann angegeben werden, ob etwa bei Auftreten zweier Units ein identisches Objekt gemeint ist oder, wie bereits angedeutet, durch einen nichtleeren Durchschnitt zweier Units davon ausgegangen werden muß, daß bestimmte Attribute beider Units nicht mehr unabhängig voneinander sind. Einer oder mehreren Units können Attribute zugeordnet werden, durch die ihr Zustand Forderungen unterworfen wird.
Die Units und ihre Attribute können schließlich selbst Anforderungen unterworfen werden, die sie erfüllen müssen ("Prädikatierung des Modells"). Damit enstehen prädikatierte Unit- Relationship- Modelle ("pUR-Modelle"), deren einfachste Beschreibung über Diagramme möglich wird.

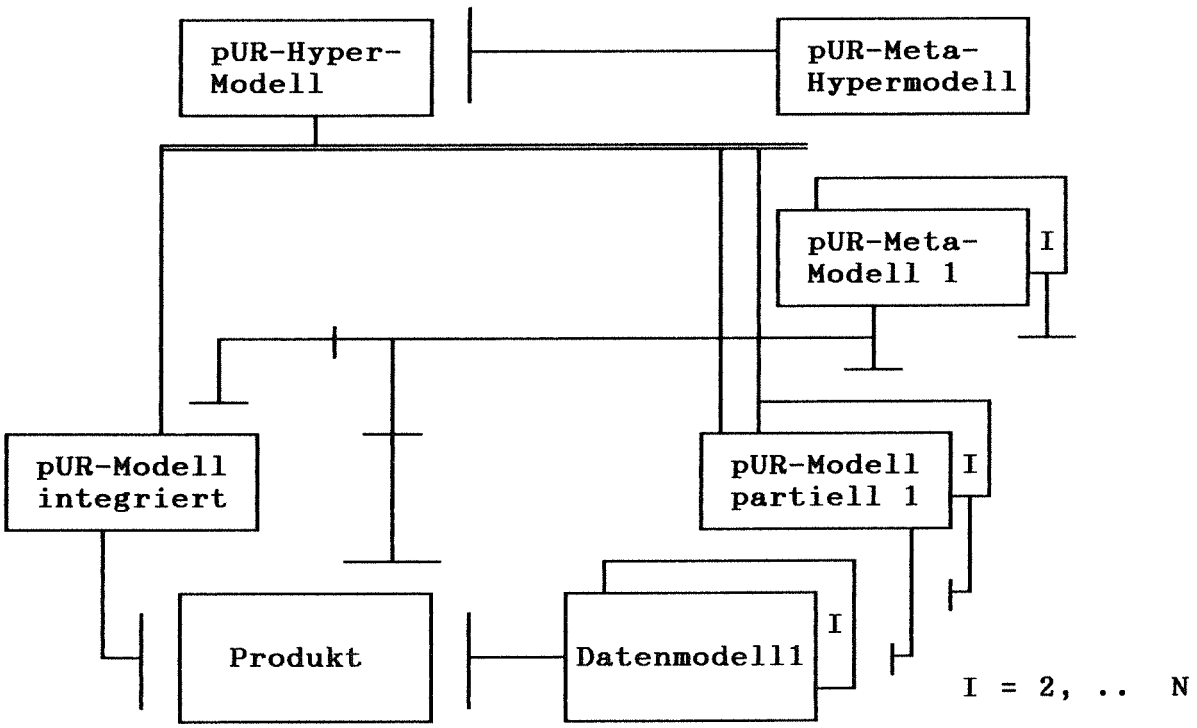


Bild 9. Zusammenhang von Produkt und Datenmodell mit pUR- Modellen

2.5. Unterschiedliche Realisierungen von Produktdatenmodellen

Die Charakterisierung eines Produktdatenmodells ist nach dem bekannten 3-Ebenen-Konzept der ANSI/SPARC -Gruppe aus dem Jahre 1976 auf

- konzeptueller
- externer
- interner

Ebene möglich. Es handelt sich hier um verschiedene Formen einer Produktbeschreibung, die

- eine menschengerechte Information
- eine rechnergerechte Ein- und Ausgabe
- eine speichergerechte Ablage

erlauben sollen. Wir behandeln sie als verschiedene physische Realisierungen von Modellbeschreibungen ein- und desselben Produkts. Je nach Datenträger wird sich die äußere Form der Darstellung unterscheiden. Im Falle eines direkt adressierbaren Speichers kann von einer Unit direkt auf eine andere Unit verwiesen werden, im Falle einer sequentiellen Datei ist das nur über einen Suchprozeß zu erreichen. Dabei wird über spezielle Attributevorgaben (z.B. Identifikatoren) eine nachträgliche Verbindung hergestellt. Die Darstellungen sind prinzipiell ineinander überführbar, ihr wechselseitiger Zusammenhang ist durch pUR-Diagramme unmittelbar darstellbar. Wir werden davon im weiteren Gebrauch machen.
Ein Datenmodell kann von dem Moment an genutzt werden, wo es im Speicher eines Rechners abgelegt ist. Die interne Darstellung muß der Anwender nicht kennen, entscheidend ist für ihn die Kenntnis der externen Repräsentation. Eine Wandlung EXTERN-INTERN erfolgt über Speicherzugriffsroutinen, die im Falle kompletter Partialmodelle prinzipiell ohne zusätzliche Nutzerinformationen abwickelbar sind. Das Ergebnis einer Ausgabe ist eine sequentielle Datei, die in vorgegebener Weise Units, Attribute, Attributeattribute usw. enthält. Umgekehrt sollte bei der Einspeicherung des Modells von einer gleichartigen Struktur des Eingabefiles ausgegangen werden können. Wesentliche Abweichungen zwischen Ein- und Ausgabeform sind jedoch im Falle eines Integrierten Produktdatenmodells zu erwarten, wo ein spezielles Partialmodell eingegeben und ein anderes Partialmodell angefordert werden kann.
Auch wenn im vorliegenden Rahmen auf die interne Darstellung von Datenmodellen nicht näher eingegangen werden kann, besteht doch eine große Wechselwirkung zwischen den Möglichkeiten einer Datenbank und dem Niveau einer Produktdatenspeicherung.Wie bereits erwähnt, werden statt der klassischen semantische Datenmodelle benötigt. Sie erfordern insbesondere eine Berücksichtigung gängiger Abstraktionskonzepte, so daß

- IS-A-Relationen ("Generalisierung")
- IS-PART-OF-Relationen ("Aggregation")
- IS-INSTANCE-OF-Relationen ("Klassifikation")
- IS-MEMBER-OF- Relationen ("Assoziation")

direkt darstellbar werden /Matt89/. Der konsequente Ausbau von Produktdatenmodellen führt letztendlich auf wissensbasierte Systeme, bei denen Prädikate in Gestalt von Regeln die Festlegung von Units und deren Attribute unterstützen. Hier existieren bereits seit langem objektorientierte Systeme (z.B. KRL 1977,FLAVORS 1982), deren prinzipielle Fähigkeiten die Anforderungen an eine Produktdatenmodellierung immer mehr beeinflussen werden /Jack87/.

3. Grundelemente von pUR-Diagrammen und ihre Bedeutung

Eine Beschreibung der Grundstrukturen von Produkten wird durch grafische Sprachen wie IDEF1X, NIAM, EXPRESS-G sehr erleichtert. Sie ermöglichen als "semantische" Beschreibungen im Gegensatz zu textlichen, "formalen" Darstellungen eine gute Veranschaulichung struktureller und topologischer Zusammenhänge. Ihre Nutzung hat eine lange Tradition, trotzdem entstehen immer wieder neue grafische Sprachen /Willi89, Dobr90/ oder eigenständige grafische Darstellungstechniken im Rahmen neuer Systeme /DGLo86, WaPi91/. Offenbar existiert noch keine Sprache, die alle Bedürfnisse befriedigt und allgemeine Akzeptanz findet.
Im folgenden soll mit der Nutzung von pUR-Diagrammen ein weitere Möglichkeit zur Diskussion gestellt werden, die Struktur allgemeiner Produkte und deren Modelle zu veranschaulichen. Dabei wird das Schwergewicht darauf gelegt, daß nicht nur eine einfach realisierbare und klare Darstellungen möglich werden, sondern die unterschiedliche Semantik der Beschreibungsdaten angemessen widergespiegelt werden kann. Gegenüber der ursprünglichen Notation /KuDo90/ gibt es eine wesentliche Weiterentwicklung, die eine schnelle Realisierung durch ein Textsystem wie Word erlaubt. Grundlage dafür ist eine konsequent objektorientierte Betrachtungsweise, bei der alle Informationen direkt oder indirekt mit klar abgegrenzten Produktteilen in Verbindung gesetzt werden. Das setzt voraus, daß beliebige Teile eines Produkts abgegrenzt werden können, deren existentieller Zusammenhang die Produktstruktur festlegt. Insbesondere sind Mehrfachbeschreibungen möglich, bei denen orthogonale Objektzerlegungen einer bestimmten Auflösung verfeinert oder vergröbert werden können. Setzen wir voraus, daß auf der untersten Ebene Basisobjekte ("Produktatome") als kleinste, nicht weiter zerlegbare Teile existieren, aus denen jedes Objekt durch Aggregation erzeugt werden kann, lassen sich klare existentielle Abhängigkeiten definieren, die zur Grundlage einer allgemeinen Strukturbeschreibung gemacht werden können. Dabei bietet das Bewußtmachen von Zwischenstadien auf dem Wege zu einer extrem einfachen Notation die Möglichkeit einer differenzierten Wiedergabe.

3.1. Units mit Attributierung

Units werden durch Kästchen, Attribute von Units durch Kreise dargestellt, die bei der Kurzform fehlen können. Jede Kästchen- Kreis- Verbindung bedeutet eine Anforderung, der eine Unit genügen muß. Sie ist automatisch erfüllt, wenn das Objekt entsprechend der Anforderung festgelegt wird. Die Bedingung ist wahr, wenn bei realer Existenz des Produkts oder Modells die prognostizierten Eigenschaften zutreffen. Damit repräsentiert eine Kästchen- Kreis- Verbindung ein Prädikat 1. Stufe, das als autonome Einheit eine klare Bedeutung hat, unabhängig vom Ort seiner Fixierung im Modell.
Die gewählte Darstellung ist sehr verbreitet, etwa im Zusammenhang mit Entity- Relationship- Diagrammen. Im Unterschied zu ER- Diagrammen, wo sich bei der praktischen Anwendung die Grenzen zwischen Objekten und Attributen teilweise verwischen, wird hier jedoch eine klare Trennung vollzogen. Einfache Kästchen stehen damit immer für

Units, d.h. Objekte realer Produkte oder Modelle ("Prädikate 0. Stufe"). Zur Verdeutlichung dieses Sachverhalts kann die Prädikatstufe 0 in der linken oberen Ecke zusätzlich mit eingetragen werden.
Offenbar sind Attribute i.a. keine statischen Größen. Ihre Ausprägung kann zeit- oder umgebungsabhängig in bestimmten Grenzen variieren. Dabei gibt es jedoch für jedes Attribut einer Unit zu einem festen Zeitpunkt immer nur einen konkreten Wert, der durch Gleichheitszeichen zugeordnet werden kann. Die Veränderlichkeit kann durch Einführung von variablen Parametern (" Attributewerten ") abgebildet werden, z.B.

Pair Pos (Nr, X, Y) | Nr = 1, X = 20, Y = 30 (Bild 10).

Zusätzlich bietet die Parametrisierung eine natürliche Basis, ähnliche Attribute in Klassen zusammenzufassen, falls unterschiedliche, quantifizierbare Ausprägungen der Eigenschaften vorliegen.
Bei geeigneter Namensgebung reicht es i.a. aus, nur Parameternamen und Parameterwerte anzutragen, z. B. X1 = 20, da Attributänderungen durch Parameteränderungen realisierbar sind. Im Grenzfall kann ein Attribut zwar von Unit zu Unit differieren, jedoch für eine Unit unveränderlich sein, etwa im Falle eines Identifikators

Identifikator (Ident) | Ident = 100 (Bild 10).

Auf einer anderen Ebene liegt die Vorgabe eines zulässigen Wertebereichs

MinimalWert < AttributeWert < MaximalWert

für ein konkretes Attribut. Sie bedeutet, daß das beschriebene Objekt noch nicht eindeutig festgelegt oder ausgewählt ist. Eine Längenvorgabe L = 10 .. 100 drückt dann nur aus, daß das Produkt am Ende eine Länge aufweisen muß, die in diesen Grenzen liegt.
In bestimmten Fällen kann ein- und dieselbe Eigenschaft auch mehrfach einer Unit zugeordnet werden. In diesen Fällen bezieht sie sich aber tatsächlich auf unterschiedliche Sachverhalte, z.B. auf verschiedene Objektteile. So läßt sich eine Gerade durch zwei Punkte charakterisieren, die dann aber Anfangs- und Endpunkt beschreiben. Für eine einfache Unterscheidung bietet sich in solchen Fällen die Indizierung im Rahmen einer Feldvereinbarung an (Bild 10).

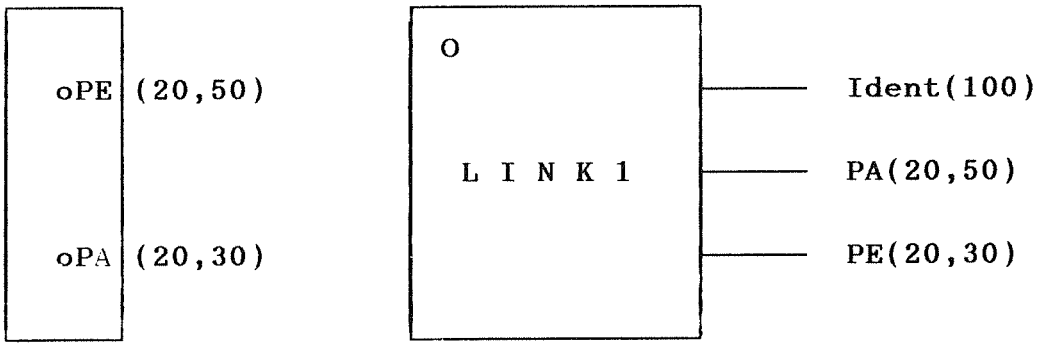


Bild 10. Darstellung einer Unit mit Attributen

Ein Sonderfall liegt vor, wenn eine Unit mit jeweils speziellen Attributen unter verschiedenen Namen mehrmals im Schema auftritt (Mehrfachpräsenz identischer Units). Eine

solche Situation ist gegeben, wenn aus einer Klasse Units entnommen werden, die verschiedene Relationen befriedigen sollen. Hier wird eine Unterscheidung wichtig, ob unterschiedliche Vertreter der Klasse oder derselbe Vertreter der Klasse die Relationen befriedigen sollen (Bild 11). Die Äquivalenz identischer Units kann durch ein Gleichheits-

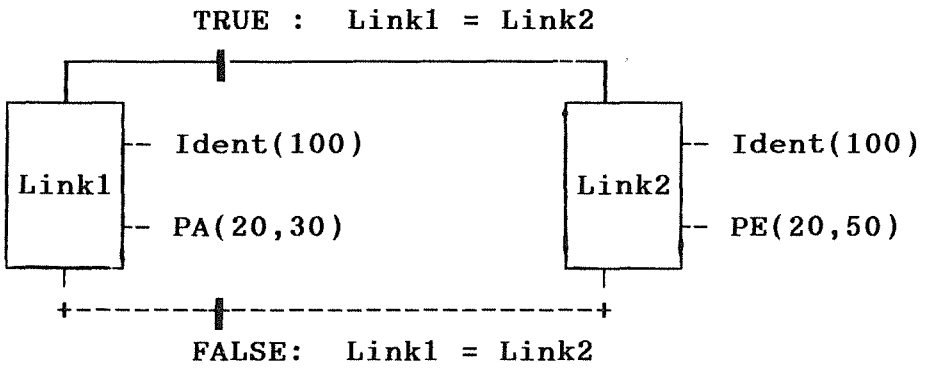


Bild 11. Mehrfachpräsenz einer Unit

zeichen verdeutlicht werden, jedoch bei entsprechender Vereinbarung entfallen. Das Gegenstück ist eine Nichtäquivalenz von Units, die durch ein besonderes Negationssymbol angezeigt werden kann.

3.2. Unitrelationen mit Attributierung

Nicht alle Attribute lassen sich eindeutig einer Unit zuordnen. Manche Attribute, z.B. der Abstand zweier Objekte, charakterisieren mehrere Units gemeinsam, ohne daß die Festlegung einer eigenen "Trägerunit" notwendig ist. In diesen Fällen werden die beteiligten Units durch einfaches Auflisten zu einer Menge zusammengefaßt, die unmittelbar zur Definition einer Unitrelation herangezogen werden kann. Die Unitrelation enthält dann

a) gemeinsame Attribute

b) zugehörige Units

c)Attribute zugehöriger Units.

Unitrelationen werden durch Relationship-Kästchen mit dünner oder strichlierter Randlinie dargestellt. Die Strichlierung soll andeuten, daß der Relation keine eigenständige Unit entspricht. Die Gleichsetzung der Unitrelation mit der Menge zugeordneter Units ("Elementeunits") erfolgt über Verknüpfung mit einer Doppellinie ("Äquivalenzschiene") von jeweils unterschiedlicher Seite aus. Sie repräsentiert eine Vorschrift, durch die die Verknüpfung der zugehörigen Elemente erfolgt. Die Äquivalenzschiene kann am Ende mit einem Verknüpfungsattribut versehen sein, um die Art der Zusammenfassung der Elementeunits näher zu kennzeichnen. Wie zuvor kann dazu ein Kreis mit Attributantrag genutzt werden, der jedoch in der Kurzform weggelassen oder durch spezielle Symbole ersetzt werden kann.

Wir vereinbaren, daß im Falle einer Vereinigung wegen der Häufigkeit des Vorkommens das Verknüpfungsattribut ganz entfallen kann oder durch ein unteres Halbkästchen symbolisch dargestellt wird. ("Vereinigungssymbol"). Die Wahl der Zeichen ist durch die Möglichkeiten des Textsystems bestimmt und nicht zwingend, wichtig ist jedoch, die Semantik in einer Weise abzubilden, daß eine Vergröberung oder Verfeinerung der Darstellung logischen Gesetzen gehorcht. Auf diese Weise lassen sich mehrstellige Attribute im Modell verankern, die eine reale Beziehung mehrerer Units charakterisieren. Das bedeutet wieder wie im Falle einfacher Units eine Behauptung, die durch Vergleich mit

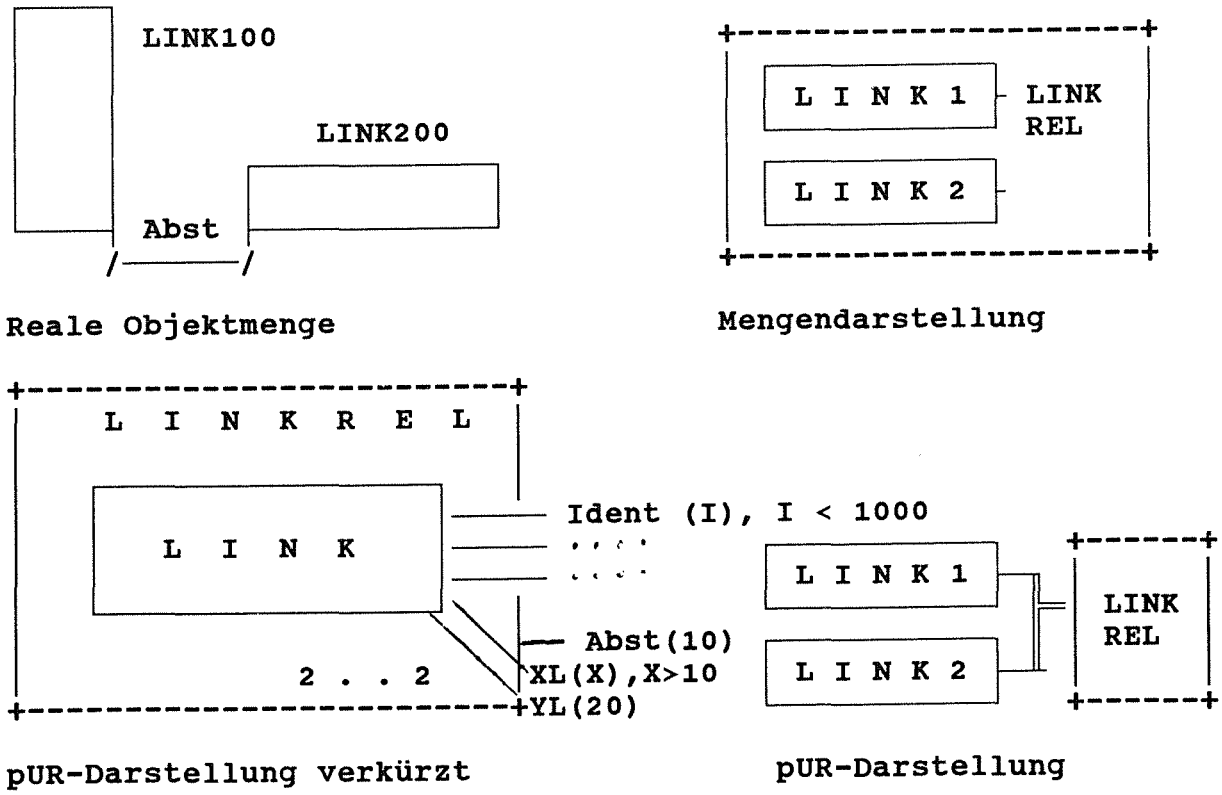


Bild 12. Verschiedene Darstellungen einer Unitrelation

der Wirklichkeit auf ihren Wahrheitswert zu überprüfen ist.
Äquivalenzen bedeuten Redundanz. Es sollte daher immer möglich sein, überflüssige Informationen wegzulassen, ohne die Klarheit zu gefährden. Dafür gibt es zwei Möglichkeiten, indem entweder die Relationship-Kästchen oder die detaillierte Elementeverknüpfung mittels Äquivalenzschiene unterdrückt werden. Bei völligem Wegfall des Relationship- Kästchens besteht die Möglichkeit, die Teilmenge zugehöriger Elemente als Seitenast einer umfassenderen Menge an die dortige Äquivalenzschiene mit anzuschließen (Bild 14). Geht es nur darum, vorhandene Units einer zusätzlichen Relation zu unterwerfen, ohne daß die Trägermenge eigenständiges Interesse beansprucht, kann das entspre-

chende Attribut auch direkt an eine jetzt als Dickstrich dargestellte spezielle "Relationsschiene" angetragen werden, die an die Stelle der Äquivalenzschiene tritt.
Im zweiten Fall ist es notwendig, auf andere Weise den Zusammenhang zu den gemeinten Elementeunits rekonstruieren zu können. Dafür kann gelegentlich die zusätzliche Einführung eines weiteren Kästchens im Innern nützlich sein, das als beliebiger Vertreter aus der Menge der zugeordneten Elementeunits zu verstehen ist. Attribute, die diesem inneren Kästchen zugeordnet werden, beziehen sich auf jedes einzelne Unitelement. Ihre eigentliche Bedeutung liegt darin, daß sie eine Grundlage für die Festlegung von Schematas liefern. Im vorliegenden Zusammenhang ist es mit ihrer Hilfe gelegentlich möglich, durch Vorgabe charakterisierender Eigenschaften ("Auswahleigenschaften") die beteiligten Elementeunits bereits hinreichend festzulegen, ohne alle Instanzen einzeln angeben zu müssen. Attribute sind die eigentlichen Einflußgrößen, durch die Objektzustände verändert werden können. Da sie im Grunde keine von Objekten unabhängige Existenz haben, werden sie gelegentlich mit den Prädikaten selbst gleichgesetzt (Bild 12) .

3.3. Vereinigungsunits mit Attributierung

Eine Unitrelation kann zu einer eigenständigen Unit ("Vereinigungsunit") gemacht werden. Das ist in allen Fällen notwendig, wo zusammengesetzte Objekte als neue Einheiten betrachtet werden, die nur durch eigene Eigenschaften charakterisiert werden, ohne daß noch deren Zerlegung in Teile und die Erfassung der Teileeigenschaften eine Rolle spielt ("Black-Box- Prinzip"). Eine solche Situation ist etwa gegeben, wenn ein und derselbe Weltauschnitt mehrmals in jeweils unterschiedlich große Objekte ("Produktgranulate") zerlegt und seperat durch direkten Vergleich mit der Wirklichkeit beschrieben wird.
Wie bei Relationship- Kästchen kann wieder ein inneres Vertreterelement festgelegt werden, an das allen Elementeunits gemeinsame Eigenschaften angetragen werden können. Attribute, die dem äußeren Kästchen zugeordnet werden, stellen Attribute der Vereinigungsunit dar, sind also durch direkten Vergleich mit der Wirklichkeit zu gewinnen und zu überprüfen (Bild 13).

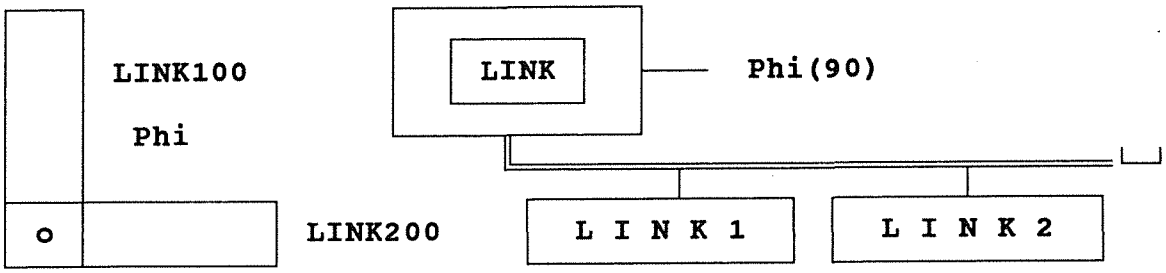


Bild 13. Darstellung von Vereinigungsunits

Units ohne Unitelemente heißen Basisunits oder (Modell/ Produkt)- Atome. Mit ihrer Hilfe wird es durch Reduzierung möglich, eine beliebige Unit durch eine Menge von Basisunits darzustellen. Dadurch wird letztendlich entscheidbar, ob Units gemeinsame Elemente besitzen oder nicht. Existieren keine gemeinsamen Elemente, sind die Units unabhängig und elementefremd, andernfalls besteht eine existentielle Abhängigkeit. Eine saubere Modellierung muß derartige Abhängigkeiten entweder vermeiden ("Orthogonali-

sierung") oder transparent machen. Im weiteren werden daher nur Units als isolierte Einheiten dargestellt, die voneinander unabhängig sind. In allen anderen Fällen ist über geeignete Äquivalenzen deutlich zu machen, wie die existentielle Abhängigkeit der Units untereinander beschaffen ist.
Die Verwendung von eigenen Namen wie LINK1 statt LINK100 soll den Unterschied zwischen Modell- und Produktunit noch einmal verdeutlichen.

3.4. Variantenunits und Unitklassen

Häufig ist es zweckmäßig, ähnliche Units nicht unabhängig voneinander zu beschreiben, sondern in einem gemeinsamen Schema. Der Vorteil dieses Verfahrens besteht darin, daß gleiche Teile nur einmal charakterisiert werden müssen. Als Nachteil ergibt sich offenbar, daß bei Verwendung einer derartigen Unit ("Variantenunit") zusätzlich Auswahleigenschaften vorgegeben werden müssen, um die tatsächlich gemeinte Unit herauszuschälen.
Unitvereinigungen bzw. Vereinigungsunits können optionale Unitelemente besitzen, die durch einen einfachen Querstrich auf der Verbindungslinie kenntlich gemacht werden. In diesem Fall kann ein und derselben Unit eine unterschiedliche Menge von Unitelementen entsprechen. Ein Sonderfall ist die Alternative, bei der zu einem bestimmten Zeitpunkt immer nur genau ein Unitelement zugeordnet werden darf und die statt der Vereinigungsmenge nur eine einzelne Elementeunit zuläßt (Bild 14a) . Auf diese Weise lassen sich Objektmengen einbeziehen, die durch unterschiedliche Komponenten flexibel zusammengesetzt werden können. An die Stelle einer festen Unitanzahl treten jetzt eine untere und obere Grenze der Anzahl (Bild 14b).

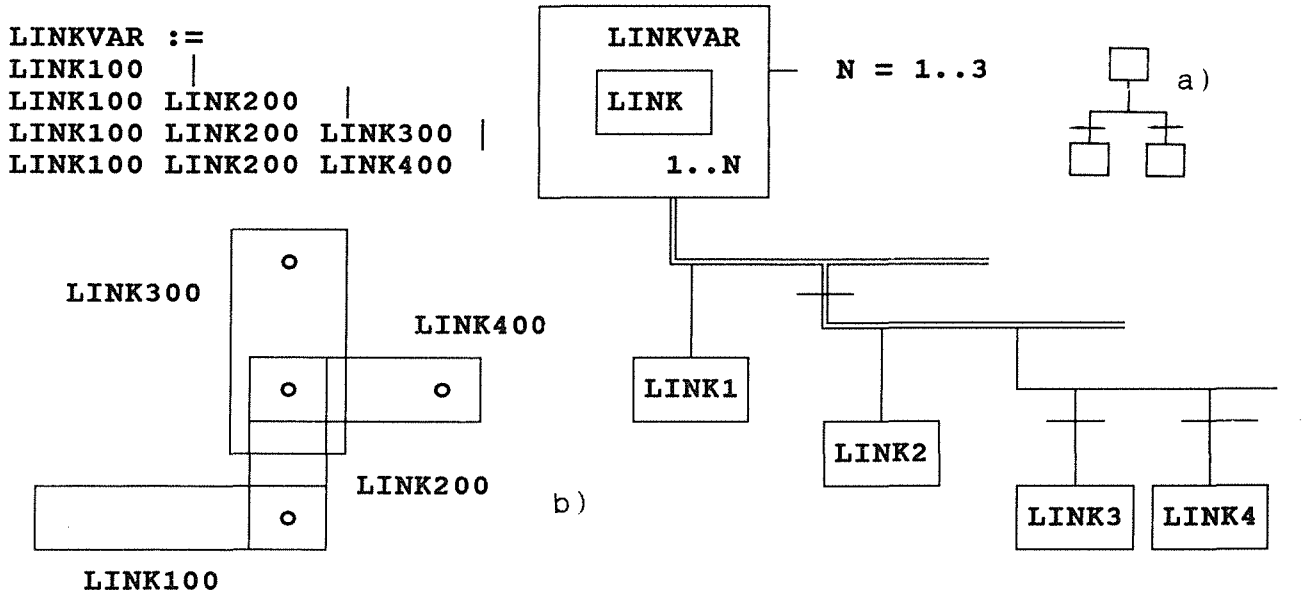


Bild 14. Verschiedene Varianten eines Roboterarms
a) Alternative Auswahl unter Elementeunits
b) Variable Anzahl von Elementeunits

Als eine besondere Form von Variantenunits lassen sich Unitklassen ansehen, mit deren Hilfe Units mit gleichem Attributeschema zusammengefaßt werden können. Durch Bereitstellung von Auswahlattributen wird es möglich, genau eine Elementeunit oder eine Teilmenge von Elementeunits auszuwählen. Auf dieser Grundlage kann eine wiederholte Nutzung von Elementeunits im gleichen oder in verschiedenen Modellen gesichert werden, ohne daß jedesmal eine Neudefinition nötig wird. Die Verwaltung derartiger Klassen von Units ist eine Domäne der klassischen Datenbanktechnologie.
Spätestens mit der Einführung von Unitklassen ensteht der Zwang, einen einheitlichen strukturellen Aufbau von Units beschreiben zu können. Es muß also möglich sein,

- der Vertreterunit eine Menge von Units zuzuordnen, die etwa über Vereinigung oder Durchschnitt mit der Elementeunit zusammenhängen
- die Zuordnung durch Auswahl aus einer Unitklasse variabel zu halten.

Besondere Bedeutung besitzt der Fall, daß in einer Oberunit mit l Elementeunits alle Unterunits eine feste Anzahl m von Elementeunits aufweisen, umgekehrt jede Elementeunit in n verschiedenen Unterunits enthalten ist. Die harten Randbedingungen können gemildert werden, wenn statt der festen Anzahl von Elementeunits zulässige Grenzbereiche festgelegt werden. Wir stoßen hier auf M:N- Beziehungen, die im Rahmen von Entity-Relationship- Diagrammen eine besondere Bedeutung erlangt haben. Werden je zwei gekoppelte Links zu einer neuen Einheit ("Joint") zusammengefaßt, ergibt sich so der in Bild 15 dargestellte Zusammenhang.

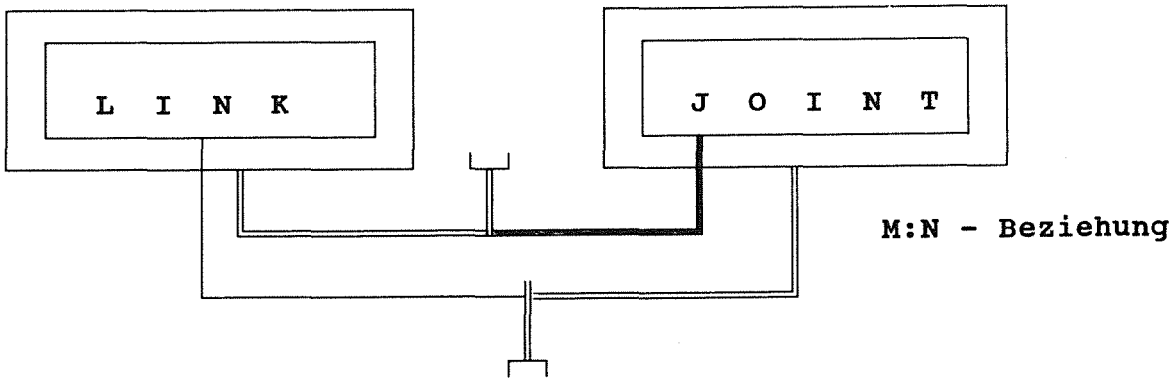
3.5. Durchschnittsunits mit Attributierung

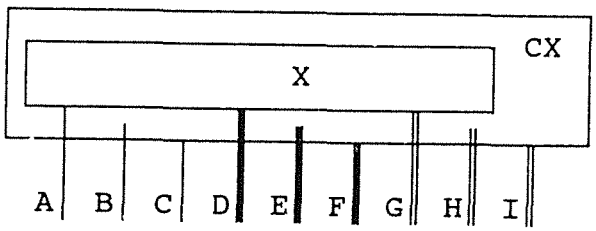
Wir haben im Zusammenhang mit den Vereinigungsunits die Forderung erhoben, daß für alle Units existentielle Abhängigkeiten kenntlich gemacht werden müssen. Das ist ohne Erfassung von Unitdurchschnitten nicht möglich. Vorausgesetzt wird dazu eine Menge gemeinsamer, unabhängiger Elemeteunits U1, U2, ... Un, mit deren Hilfe über unterschiedliche Vereinigungsmengen V1, V2, ... Vn die zu überprüfenden Units erzeugbar sein sollen. Mit ihrer Hilfe läßt sich klären, ob die Units einen gemeinsamen Durchschnitt besitzen oder nicht. Es entsteht eine Durchschnittsunit D, deren Größe unmittelbar von der Wahl von V1, V2, .. Vn abhängt. Die Verkleinerung einer einzigen Unit Vi, i = 1, 2,... n kann bereits eine Verkleinerung von D nach sich ziehen, umgekehrt bedeutet eine Erweiterung aller beteiligten Units V1, V2, ... Vn um dieselbe Elementeunit auch eine Erweiterung von D.
Es ist besonders zu beachten, daß sich bei dieser Bestimmung die Elementeunits verkleinern können. Die Durchschnittsunit kann damit eine andere Elementebasis haben als die Ausgangsunits. Derartige Unterschiede entfallen erst auf atomarer Ebene, die aus unteilbaren Units gebildet wird und mit deren Hilfe beliebige Units durch Vereinigung zusammengesetzt werden können.
Die Ermittelung von Durchschnittsbeziehungen hat besondere Bedeutung für eine Strukturanalyse. Eine Verwendung von Units mit nichtleerem Durchschnitt bedeutet, daß diese Units nicht unabhängig voneinander sind, sondern über gemeinsame Elementeunits verkoppelt sind. Eine Veränderung von Eigenschaften innerhalb einer Unit kann also Auswirkungen auf eine andere Unit haben.
Es gelten folgende Aussagen:

- Für jede Unit LINK existieren mehrere (= 2) Joints, deren Durchschnitt äquivalent zur Unit LINK ist.

- Für jede Unit JOINT existieren mehrere (= 2) Links, durch deren Vereinigung die Unit JOINT entsteht.

Allgemeinere Zusammenhänge lassen sich mit Hilfe der WORD - Pseudografik einfach wie folgt ausdrücken:





Für jede Unit X der Unitklasse CX
- gilt die Beziehung A
- gilt für alle Elementeunits von X die Beziehung D
- gilt für einige Elementeunits von X die Beziehung G

Es gibt eine Unit X der Klasse CX, für die
- die Beziehung B gilt
- alle Basiselemente von X die Beziehung E erfüllen
- einige Basiselemente von X die Beziehung H erfüllen

Für die Unitklasse CX als Unit
- gilt die Beziehung C
- gilt für alle Units X die Beziehung F
- gilt für einige Units X die Beziehung I

Bild 15. Darstellung von Unitklassen

3.6. Spezielle Unitstrukturen

Die Struktur einer Unit U oder Unitvereinigung V wird durch die mehrstelligen Attribute ihrer n Elementeunits U1, U2, ... Un geprägt. Wichtige Grenzfälle liegen vor, wenn je zwei dieser Attribute keine gemeinsame Elementeunit Ui (i = 1, 2, ..,n) besitzen ("Unabhängige Attribute") oder alle Attribute auf der vollen Menge der Unitelemente U1, U2, ... Un erklärt sind ("Attributesystem über n Units"). Werden die zusammengehörigen Trägerunits Ui1, Ui2, ... Uim (m < = n) mehrstelliger Attribute zu eigenständigen Units ("Unterunits") gemacht, so sind diese Unterunits dann innerhalb der Unit U ("Oberunit") entweder paarweise durchschnittsfremd ("Orthogonalität") oder repräsentieren sämtlich dieselbe Unit wie die Oberunit ("Identität"). Offenbar läßt sich eine Vereinigung orthogonaler

Units ("Aggregation") wegen des Fehlens existentieller Abhängigkeiten besonders einfach handhaben.

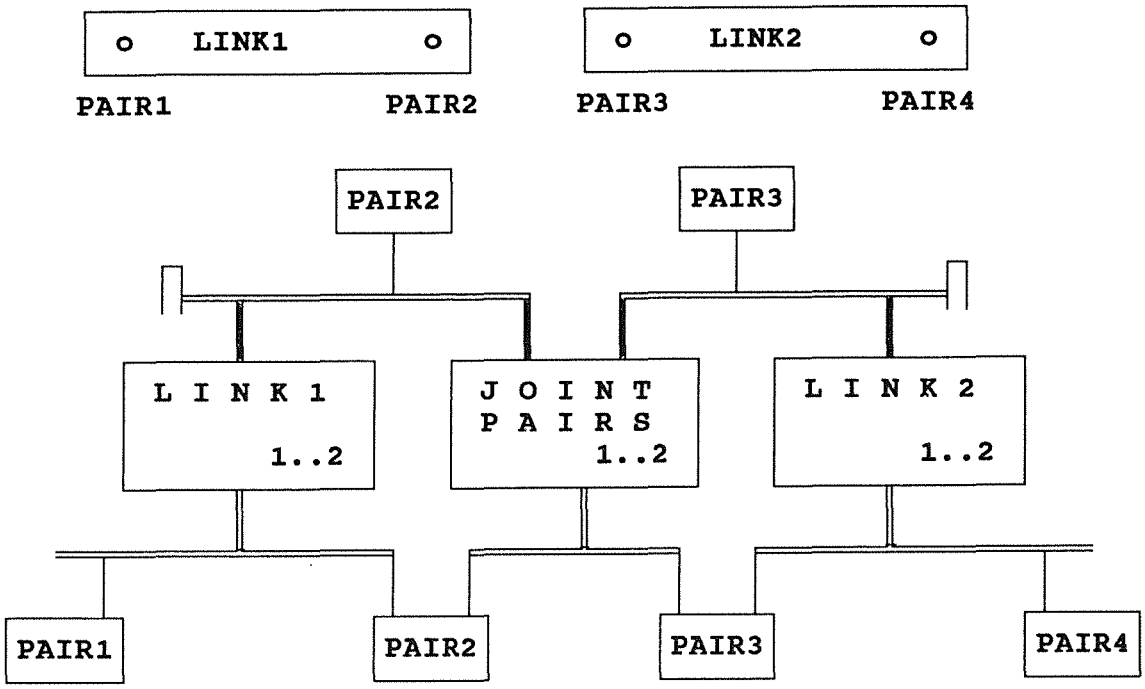


Bild 16. Durchschnitt von Units LINK1 bzw. LINK2 mit JOINTPAIRS

Ein regelmäßiger Strukturaufbau einer Unit beruht auf einem regelmäßigen Zusammenhang ihrer Unterunits bzw. der zugehörigen Elementeunitvereinigungen (Bild 16).

3.7. Prädikatierung von Unitattributen

Unitattribute selbst können durch eigene Prädikate ("Modellprädikate 1. Stufe") festgelegt oder eingegrenzt werden. Betrachten wir ein und mehrstellige Unitattribute als Objektprädikate 1. Stufe, können die Modellprädikate auch als Objektprädikate 2. Stufe angesehen werden. Die Verbindung zu den charakterisierten Units wird durch Verbindungslinien hergestellt, die an der Ober- oder Unterseite der Unitkästchen mit Querstrich enden. Den Prädikaten können wieder wie im Falle der Units ein- oder mehrstellige Attribute ("Attributeattribute") mit Kreisdarstellung zugeordnet werden. Bei erneuter Zuordnung von Prädikaten lassen sich auf diese Weise rekursiv Prädikate beliebiger Stufe (unter evtl. Antrag der Stufennummer) darstellen (Bild 17).
Prädikate sind überall dort von Bedeutung, wo eine direkte Auswahl oder Veränderung von Produkteigenschaften zulässig sind, die Attributewerte also nicht automatisch festliegen. Hier kann es bei Eigenschaften mit hoher Änderungshäufigkeit eine große Hilfe sein, wenn fehlerhafte Änderungen bemerkt und behandelt werden können. Die Fixierung derartiger Zwangsbedingungen wird immer ein Kompromiß bleiben, da eine Überwachung aller bekannten Einschränkungen letztendlich auf die Schaffung eines Expertensystems und eine Wissensverarbeitung hinausläuft. Ein anschauliches Beispiel für ihre Bedeutung liefert bereits die Verkopplung zweier zunächst unabhängiger Links durch Nullsetzen des Abstands ihrer Koppelstellen.

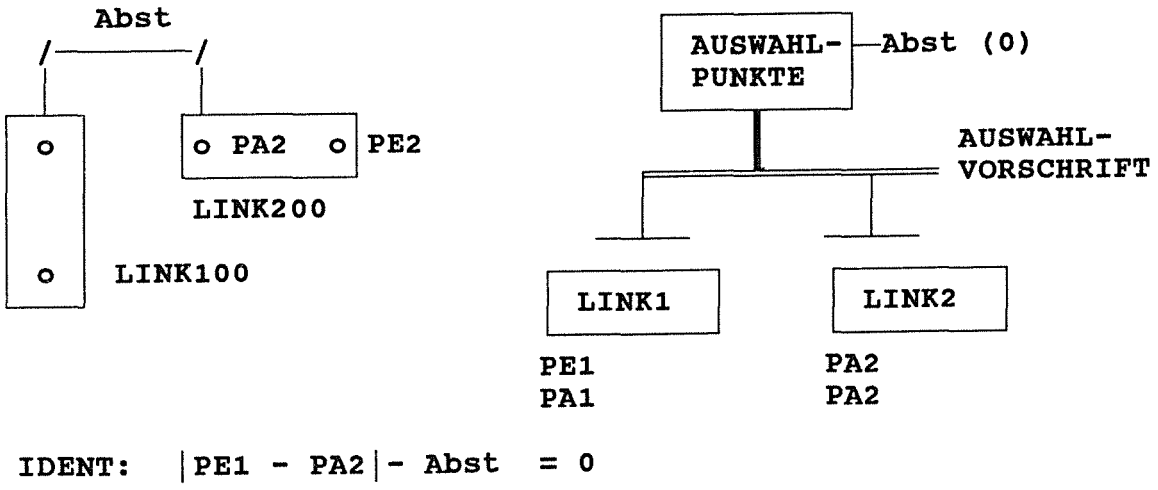


Bild 17. Prädikatierung zweier Units

3.8. Funktionelle Abhängigkeiten

Besondere Bedeutung besitzen unter den mehrstelligen Attributen funktionelle Abhängigkeiten, bei denen der Wert eines Attributs durch die Werte der restlichen Attribute vollständig festgelegt ist. In der Menge aller an einer Beziehung beteiligten Attribute bilden die funktionell abhängigen Attribute eine Teilmenge, die wieder in besonderer Weise markiert wird. Zur Veranschaulichung greifen wir noch einmal auf das Beispiel aus dem Abschnitt 2.1 zurück und nehmen vereinfachend an, daß zu jedem Link der Abstand der Pairs und der Drehwinkel in einem gemeinsamen absoluten Bezugssystem bekannt ist. In diesem Fall läßt sich die Werkzeugposition, gleichgesetzt mit PE3, sofort durch eine Funktion berechnen, sofern PA1 bekannt ist (Bild 18).
Sind keine Mißverständnisse zu befürchten, kann vereinfachend eine horizontale Verknüpfung der Trägerunits genutzt werden. Das gemeinsame Endstück der Linien verweist auf die Unit mit dem abhängigen Attribut und schließt genau hier mit Querstrich ab. Die Darstellung korrespondiert gut mit der natürlichen Vorstellung von Abhängigkeit, derzufolge Zustände von Objekten Einfluß auf Zustände anderer Objekte nehmen können (Bild 18).
Die funktionellen Abhängigkeiten schlagen eine Brücke zur Welt der Programmiersprachen. Die Reihenfolge ihrer Abarbeitung kann explizit oder implizit festgelegt werden. Im letzteren Fall entscheidet über die Aufrufbarkeit die Existenz aller benötigten Quellattribute (Petri-Netze) oder der Wahrheitswert einer Bedingung, der in der Regel durch Vergleich eines Attributewertes mit bestimmten Grenzwerten einfach ermittelt wird (Programmverzweigung).
Es ist jederzeit möglich, die Grundelemente von pUR-Diagrammen auch in formaler Notation zu formulieren. Die Darstellung wird dadurch jedoch nicht exakter, sondern für den Anwender nur schwerer verständlich.

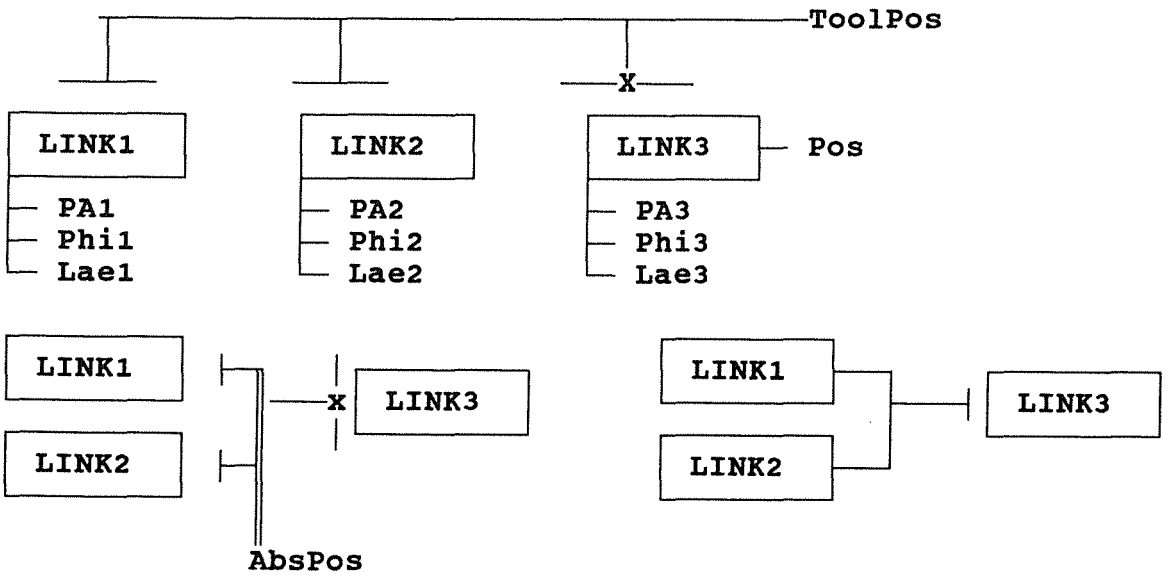


Bild 18. Verschiedene Darstellungen funktioneller Abhängigkeiten

3.9. Formalisierung von pUR-Diagrammen

pUR-Diagramme orientieren sich an den Fähigkeiten des Menschen, ihre Umsetzung auf dem Rechner erfordert eine Wandlung der Darstellungsform. Bewährt haben sich Datenschemata, durch die für Datenmengen einheitlichen Aufbaus ein Datentyp deklariert werden kann. Sie bilden in STEP die Grundlage für eine Festlegung von Entities. Konkrete Units, Unitrelationen, Attribute, Attributerelationen sind dann Ausprägungen (Instanzen) des jeweiligen Datentyps (Entities), wobei durch gemeinsam einzuhaltende Zusatzbedingungen (z.B. Anfangswerte, Wertebereich) eine nähere Spezifizierung möglich ist. Relationen erfordern dabei eine Auswahl unter möglichen Units bzw. Attributen, die durch direkte Referenzierung, Nutzung von Mengenfunktionen (z.B. Vereinigung, Durchschnitt) oder Vorgabe von Auswahlattributen, z.B. von Identifikatoren, praktikabel gemacht werden kann.
Die Festlegung von Units und Attributen erfolgt willkürlich durch Zuordnung zu Objekten und Eigenschaften der Realität . Ganz anders sieht das aus, wenn Units (und Unitrelationen) mit Attributen verknüpft werden (Prädikatierung). Das geschieht im einfachsten Fall dadurch, daß das Schema der Unit um entsprechende Attribute vergrößert wird. Bestimmte Objekte können nur bestimmte Attribute besitzen, so daß unwahre Konstellationen möglich werden. Prädikate stellen sich damit als Relationen dar, die Units und Attribute, Attribute/ Units und Attributeattribute usf. in Beziehung zueinander setzen und die wahr oder falsch sein können. Eine Gleichsetzung von Attributen und Prädikaten ist dabei in dem Sinne möglich, daß eine Relation

Rel1(Attr1, Unit2) in Attr1(Unit2)

umgewandelt wird, bei der nicht mehr Attr1 und Unit2 , sondern nur noch Unit2 geeignet zu wählen ist, um die Anforderungen zu befriedigen. Auf dieser Grundlage lassen sich Units, Attribute, Attributeattribute, ... als Prädikate 0., 1. ,2. ... Ordnung betrachten, für deren Kennzeichnung etwa die Erfassung einer Stufennummer von prinzipieller Bedeutung ist.
Wird auf eine derartige Klassifizierung der Daten verzichtet, müssen die Beziehungen zwischen Units auf Beziehungen zwischen den zugehörigen Attributemengen abgebildet werden. Das erfordert etwa bei verschiedenen Partialmodellen zum gleichen Weltausschnitt eine wechselseitige Referenzierung jener Attributemengen, die identischen Objekten zugeordnet sind. Auch wenn durch eine besondere Verarbeitungsstrategie von Partialmodellen, etwa die generelle Nutzung eines Partialmodells als Grundmodell eines Anwendungsmodells, der diesbezügliche Aufwand noch einmal reduziert werden kann, läßt sich eine Erfassung derartiger Zusammenhänge nicht vermeiden, soll eine Integration der verschiedenen Partialmodelle sinnvoll unterstützt werden. Fehlen solche Zuordnungen, kann in den einzelnen Partialmodellen nur noch die Existenz von Objekten zu bestimmten Attributemengen festgestellt werden, ohne daß deren Übereinstimmung oder Nichtübereinstimmung mit Objekten außerhalb des jeweiligen Partialmodells erschlossen werden kann. Eine Berücksichtigung von Prädikatestufen könnte den Weg zu einer offenen, erweiterungsfähigen Systemarchitektur wesentlich erleichtern.

4. Über Produktbeschreibungssprachen und Sprachübersetzer

Trotz Vordringens der Grafik spielt auch heute noch die Sprache bei der externen Darstellung von Produktdatenmodellen die entscheidende Rolle. Eine solche Darstellung ist je nach Wahl der sprachlichen Mittel auf vielfältige Weise realisierbar. Daraus resultiert zwangsläufig das Problem, ob und mit welchem Aufwand eine wechselseitige Übersetzbarkeit möglich ist. Für eine Lösung ist der Zusammenhang eines produktbeschreibenden Textes mit dem beschriebenen Produkt (oder einem äquivalenten Integrierten Produktdatenmodell) wesentlich. Er legt die Semantik des Beschreibungstextes fest und muß damit in anspruchsvolleren Fällen bekannt sein, soll eine korrekte Übersetzung des Textes und nicht nur eine Ersetzung von Worten gewährleistet werden.

4.1. Der Zusammenhang von Produktdatenmodellen und Produktbeschreibungssprachen

Die sprachliche Beschreibung eines Produkts benutzt Zeichenfolgen. Voraussetzung für deren sinnvollen Gebrauch ist eine Grammatik , mit deren Hilfe

- zulässige Zeichen ("Alphabet")
- zulässige Wörter ("Terminals") und
- zulässige Sätze (" Nichtterminals")

allgemeingültig charakterisiert werden. Dabei muß die Notation nicht notwendig in Klartext, d.h. für den Menschen lesbar, vorliegen. So sollten Zahlen in maschineninterner Form bei Austausch auf Maschinenebene direkt übertragen werden können, ohne daß jedesmal eine Konvertierung in lesbare Zeichen erfolgen muß.
Betrachten wir die physische Realisierung eines Textes, so liegt ihr ein Datenträger wie Papier oder Diskette zugrunde, der sich auf eine lineare Folge von Textfeldern zurückführen läßt. Jedes der Textfelder fester oder variabler Länge kann durch Eintrag von Zeichen in unterschiedliche Zustände gebracht werden. Wert, Datentyp oder Länge stellen dann Attribute dar, die die Textfelder näher charakterisieren. Offenbar kann der Datenträger als ein konkretes, universell einsetzbares Modell gesehen werden, dessen Units aus zusammenhängenden Textbereichen unterschiedlicher Größe bestehen. Durch schrittweise Zusammenfassung benachbarter Textfelder läßt sich eine Hierarchie von Units definieren. Die Zeicheneinträge entsprechen dann den Terminals, die Units selber Nichtterminals der zugehörigen Grammatik.
Jeder Text kann nun als eigenständiges Modell durch ein pUR-Schema beschrieben werden. Da für das zu modellierende Produkt in gleicher Weise eine pUR-Darstellung möglich ist, existiert dann eine gemeinsame Plattform, um durch wechselseitiges Zuordnen der Units, Attribute, Attributeattribute usw. die Semantik des Textes festzulegen. Im Gegensatz zu pUR-Diagrammen und auch im Gegensatz zu einem konkreten Produkt sieht man einer Zeichensequenz jedoch nicht ohne weiteres an, ob sie ein Objekt oder eine Eigenschaft einer Eigenschaft widerspiegelt und zu welchem Objekt eine Eigenschaft gehören soll. Erreichbar wird das erst durch zusätzliche Informationen, die im Normalfall eine bestimmte Reihenfolge oder weitere unterscheidende Textzeichen erforderlich ma-

chen. Mit Hilfe derartiger Zusatzmerkmale ist dann auch hier entscheidbar, ob Daten Objekte, Objektattribute oder Attribute von Objektattributen beschreiben und zu welchen Objekten Attribute jeweils gehören. Auf dieser Grundlage kann dann eine Verbindung zur Wirklichkeit hergestellt werden, die überprüfbar wird. Eine Modellunit, die unter Bereitstellung von Attributen ein Objekt repräsentieren soll, würde die Existenz eines entsprechenden realen Objekts mit entsprechenden Eigenschaften behaupten. Das wäre jedoch eine Aussage mit Wahrheitswert, die auch im herkömmlichen Sinne Satz einer Sprache ist.

Im ISO-Standard-Entwurf STEP für den Austausch von Produktdatenmodellen werden die Datensätze der physischen Datei als Entities bereitgestellt. Entities beinhalten Zusammenstellungen konkreter Attribute und konkreter Entity-Referenzen. Entities ähnlicher Struktur können unter Zuordnung eine Typbezeichnung klassifiziert und unabhängig davon durch Identifikatoren als Einzelexemplare unterscheidbar gemacht werden. In STEP werden auf der Dateiebene Aussagen der Form

'Das Entity vom Typ JOINT mit dem Identifikator 100 greift auf 2 Entities (vom Typ LINK) mit den Identifikatoren 111 und 112 und ein Entity (vom Typ PAIR) mit dem Identifikator 113 zurück'

durch die Notation

#100 = JOINT (#111, #112, #113);

ausgedrückt. Die Referenzierung der Entities 111, 112 und 113 bedeutet eine Erweiterung um deren Daten. Sie ermöglicht eine Vernetzung der Entities dergestalt, daß die Daten eines referenzierten Entities gleichzeitig in verschiedenen anderen Entities verwendet werden können. Dabei fehlt jedoch eine direkte Aussage, ob etwa die Entities 100 , 111, 112 und 113 Objekte, Attribute oder Attributeattribute des Produkts repräsentieren und wie etwa der existentielle Zusammenhang der referenzierten Objekte (z.B. Oberobjekt, Unterobjekt) ist.
Für eine weitere Arbeit mit den Modellen kann die Fixierung der unterschiedlichen Bedeutungen der Daten nur von Nutzen sein. Das zeigt sich bereits an den unterschiedlichen Konsequenzen einer Löschoperation für Objekte oder Attribute. Wird ein Objekt durch Löschen des zugehörigen Entities beseitigt, sind alle Attribute des Objekts mit zu tilgen. Wird ein Attribute-Entity beseitigt, kommt darin nur zum Ausdruck, daß für das zugehörige Objekt die Attribute unbestimmt werden.
Ein weiteres Problem ist die Verwendung von Entity-Identifikatoren, ohne daß eine direkte Zuordnung zu einem bestimmten Objekt entsprechenden Typs daraus abgeleitet werden kann. Unterschiedliche Partialmodelle können Units mit unterschiedlichen Identifikatoren verwenden, die ein- und dasselbe reale Objekt charakterisieren. Dieser Zusammenhang läßt sich, wird auf eine problematische Querverbindung zwischen den Partialmodellen verzichtet, nur indirekt erschließen. Einen Weg dazu eröffnet etwa eine gemeinsame Nutzung von Submodellen, so daß etwa ein Entity eines Geometrie-Submodells von verschiedenen Partialmodell-Entities aus referenziert werden kann.
Verschiedene Systeme nutzen unterschiedliche Sprachmodelle. Die wechselseitige Überführung ihrer Texte erfordert Übersetzer. Dabei sind zu Units eines Quellsprachenmodells nach einer Analyse Units in der Zielsprache zu generieren, deren Attribute aus den Attributen des Quellsprachenmodells abzuleiten sind. Derartige Zusammenhänge erfordern jedoch eine Kenntnis der Unit-Bedeutungen, die ein Wissen um den Zusammenhang mit dem zu beschreibenden Produkt voraussetzen. Der natürliche Weg für eine adäquate Übersetzung wäre der, die Auswirkungen der Units des Quellmodells auf die zugeordneten

Objekte zu verfolgen, um in einem zweiten Schritt die Units des Zielmodells daraus herzuleiten. Der Umweg über die Realität wäre zu vermeiden, wenn ein Integriertes Produktdatenmodell an die Stelle des Produkts treten würde. In diesem Fall wäre die Übersetzung ein zweistufiger Prozeß, der voll auf dem Rechner ablaufen kann.

4.2. Die Beschreibung von Schnittstellen zwischen Produktbeschreibungssprachen

Eine natürliche Grundlage der Schnittstellenbeschreibung ist die Festlegung des Zusammenhangs zwischen Quell- und Zielentities. Dazu ist je Zielentity festzustellen, welche Quellentities Trägerobjekte enthalten, die mit den Trägerunits der Zielunit einen gemeinsamen Durchschnitt aufweisen. Offenbar kann man sich bei der Festlegung der Attribute der Zielunits auf diese Menge von Quellunits stützen. Notwendig ist die Kenntnis der Abhängigkeit der Zielunitattribute von den Quellunitattributen, für die eine klare Rechenvorschrift benötigt wird.
Die Strukturierung der Quell- und Zielunits ist offenbar entscheidend dafür, wie hoch der Übersetzungsaufwand ausfällt. Wir nehmen im weiteren an, daß die kleinsten unabhängigen Trägerobjekte der Quellentities, also die Produktatome, so gewählt seien, daß durch deren Vereinigung jedes Trägerobjekt des Produkts einer möglichen Zielunit erzeugt werden kann. In gleicher Weise ist die Zusammenstellung einer Menge unabhängiger Basiseigenschaften des Produkts sinnvoll, um beliebige Modellattribute in Verbindung mit derartigen Basisattributen festlegen zu können. Trotz dieser Einschränkungen bleiben im allgemeinen Fall jedoch noch sehr komplexe Abhängigkeiten möglich, deren Behandlung den vorliegenden Rahmen sprengen würde. Immerhin gibt das Bild 19 eine leise Ahnung davon, mit welchen Problemen bei unabgestimmter Festlegung von Produktbeschreibungen ein Übersetzer fertig werden muß.
Wir wollen uns im weiteren auf die Behandlung eines einfachen Spezialfalls beschränken. Dazu setzen wir voraus, daß die Zielentities Trägerobjekte beschreiben, die durch Vereinigung der Trägerobjekte der Quellentities entstehen. In diesem Fall ist zu erwarten, daß die Attribute der Zielentities in übersichtlicher Weise aus den Attributen der Quellentities abgeleitet werden können.Die Berechnung der Attribute eines Zielentity kann nun im Rahmen einer rekursiven Funktion erfolgen, die auf die zugehörigen Quell-Entities und alle von diesen referenzierten Entities angewendet wird. Dabei kann bei der Referenzierung im allgemeinsten Fall eine Auswahl unter Entities unterschiedlichen Typs möglich sein, so daß eine entsprechende Anzahl Berechnungsfunktionen vorzusehen ist. Als einfachst denkbarer Sonderfall ist darin eine Konvertierung enthalten, bei der Zeichen unterschiedlicher Alphabete einander zugeordnet werden oder Zahlendarstellungen als Zeichenfolge in eine maschineninterne, verarbeitungsgerechte Darstellung (und umgekehrt) überführt werden.

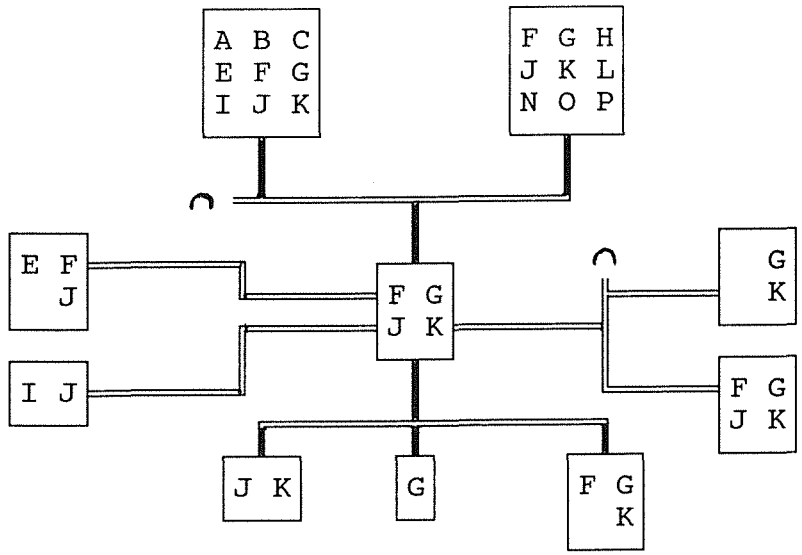


Bild 19. Existentieller Zusammenhang verschiedener Units über gleicher Grundmenge von Basisunits

An dieser Stelle wird noch einmal anschaulich deutlich, wie wichtig eine saubere Zuordnung der Attribute zu Modellunits ist, deren Trägerobjekte bei Verkleinerung die Eigenschaft nur noch unvollständig, bei Vergrößerung partiell abzuleiten gestatten. Eine unnötige Anbindung an Modellunits höherer Stufe bringt eine Reihe zusätzlicher Unterunits mit sich, die als potentielle Informationsquellen bei einer rekursiven Attributeberechnung mit durchlaufen oder wenigstens überprüft werden müssen, ohne daß sie auf das Ergebnis der Attributebestimmung tatsächlich Einfluß haben.

4.3. Eine Generierung von Übersetzern auf der Grundlage von Schnittstellenspezifikationen

Ein Übersetzer hat die Aufgabe, eine Sequenz von Quell-Entities in eine Sequenz von Ziel-Entities zu überführen, wobei Fehler erkannt und mindestens angezeigt werden sollen. Wie wir gesehen haben, kann der Zusammenhang der Entities über den gemeinsamen Rückgriff auf das modellierte Produkt (oder ein äquivalentes Integriertes Produktdatenmodell) hergestellt werden. Der Übersetzungsprozeß erfordert zunächst eine Analyse der Quell-Entities, bevor in einer Synthese die Ziel-Entities erzeugt werden. Für Quell- und Zielsprache ist dazu eine Grammatik zugrundezulegen, die Aufbau und Reihenfolge der Entities näher spezifiziert. Bei ihrer Kenntnis lassen sich mittels Scanner die zulässigen Datenworte der Quelldatei bestimmen und mittels Parser zu größeren sinntragenden Einheiten zusammensetzen. Bei der Synthese sind aus Kenntnis der Quell-Entitystruktur heraus neue Entities zu generieren, die in Übereinstimmung mit der Forderung der Zielsprachengrammatik stehen. Die Ziel-Entities beschreiben dabei mit den Mitteln der Zielsprache Produkteigenschaften, deren Herleitung aus der Produktbeschreibung mittels Quellsprache erfolgen muß.
Voraussetzung für eine Codeerzeugung ist im allgemeinen, daß die Informationen der zusammengehörigen Quellentities vollständig vorliegen, bevor eine Generierung des zugehörigen Zielentities erfolgt. Auch bei Einsatz eines Codegenerierungs-Werkzeugs kommt der Anwender nicht darum herum, die Abhängigkeit der zu erzeugenden Zielattri-

bute selbst durch Bereitstellung entsprechenden Programmcodes zu beschreiben. Ihre Implementierung ist am einfachsten mit den Mitteln der Target-Sprache möglich, die das Werkzeug zur Generierung des Zieldatei selbst benutzt.
Der Vorzug derartiger Werkzeuge besteht jedoch darin, daß eine Zuordnung des Programmcodes zu den Strukturknoten der Quellsprachengrammatik möglich wird, so daß an einen in der Synthesephase erzeugten Syntaxbaum der Quelldatei der Generierungscode für die Zielentities angefügt werden kann. Dabei läßt sich durch Verwendung einer attributierten Grammatik der Code über Parameter modifizieren.

5. STEP-Kinematikmodell als Demonstrationsbeispiel

Das Problem der Wandlung unterschiedlicher Partialmodelle ein und desselben Anwendungsgebiets läßt sich gut am Beispiel des Kinematikmodells behandeln. Am Institut für Reaktorentwicklung IRE des KfK wurde mit ROBOT ein Programmpaket entwickelt, das unter Nutzung des CAD- Systems Applicon Bravo3 eine Definition kinematischer Ketten und hierarchischer Umgebungsmodelle gestattet /LeKL91/. Die Weiterverarbeitbarkeit seiner Daten, z.B. zum Zwecke einer Bewegungssimulation unter Verwendung der Hochleistungsgrafik einer Silicon Graphics Workstation mittels KISMET /LeKL91/wird wesentlich durch die Erzeugbarkeit einer Menge von Übergabefiles bekannter Struktur unterstützt. Mit der Vorbereitung eines STEP-Kinematikmodells ist der Zeitpunkt absehbar, zu dem kinematische Ketten auf andere Weise über ein einziges STEP-Datenfile ausgetauscht werden sollen. Die bisherige ROBOT-KISMET-Schnittstelle kann unter dieser Voraussetzung nicht mehr weiterverwendet werden, wenn nicht eine Filewandlung vorgenommen wird. Eine solche Situation ist vielerorts typisch. Im Rahmen des ESPRIT-Projekts NIRO wurde vorab eine Subset des STEP-Kinematikmodells festgelegt, die eine gute Grundlage für eine experimentelle Überprüfung von Grenzen und Möglichkeiten einer automatisierten Generierung eines NIRO-ROBOT-Compilers eröffnet.

5.1. Problemstellung

NIRO und ROBOT realisieren auf unterschiedliche Weise ein Kinematikmodell. Beide Sprachen beschreiben sequentielle Dateien ("Files"), die aus Folgen von Datensätzen bestehen. Die Datensätze realisieren physisch Modellentities. Die Struktur der Files ist durch zwei Grammatiken charakterisierbar, die auf unterschiedliche Strukturbäume von Terminals und Nichterminals führen. Auf dieser Grundlage können so unterschiedliche Sachverhalte wie

- eine Vereinigung von Objekten
- eine Zuordnung von Eigenschaften zu Objekten
- eine Erweiterung der Objektbeschreibung um neue Eigenschaften durch Referenzierung anderer Entities

beschrieben werden.
Für beide Modelle bildet die reale Objektstruktur eines kinematischen Mechanismus eine natürliche Bezugsebene. Es kann gezeigt werden, daß für alle Entities eine klare Zuordnung zu Objekten besteht. Durch unterschiedliche Objektzerlegung, Attributeauswahl und Relationenfestlegung kann daraus unmittelbar eine NIRO- bzw. eine ROBOT-Darstellung abgeleitet werden.
In der Praxis stellt sich die Frage etwas anders. Gegeben ist ein Modell in einer Quellsprache, aus dem ohne Rückgriff auf die Objektstruktur ein Modell in der Zielsprache abgeleitet werden soll. Im konkreten Fall ist ein NIRO-Modell gegeben, aus dem möglichst ohne Informationsverlust ein ROBOT-File zu erzeugen ist.

Dazu sind
1. der Zusammenhang der Modelleinheiten aufzuklären und

2. der Zusammenhang der Attribute festzustellen.
Der gewünschte Compiler muß dann imstande sein, dieses Wissen für eine automatisierte Generierung des Zielfiles zu nutzen.

5.2. Strukturanalyse mittels GMD-Tools

Die Tool-Box enthält eine Reihe von Werkzeugen, die in allen Phasen der Analyse wirksame Unterstützung geben können.
1. Der Scanner Rex ("Regular Expression Tool") wird nach der Methode der regulären Ausdrücke spezifiziert. Für jedes erkannte Wort (Token) kann eine semantische Aktionvereinbart werden. So werden automatisch die Zeilen- und Linienposition für jedes Token bereitgestellt. Seine Arbeitsgeschwindigkeit ist hoch.
2. Der Parser Lalr nutzt als Spezifikation die konkrete Syntax einer kontext-freien Grammatik. Sie wird in EBNF-Notation beschrieben und gehört, wie der Name sagt, zur Klasse der LALR(1)-Grammatiken. Der Parser ist mehr als doppelt so schnell wie ein durch das bekannte UNIX-Werkzeug YACC generierter Parser.
3. Ast ("Abstract Syntax Tree") ist ein Generator für einen attributierten Syntaxbaum, der zugleich die Zugriffsroutinen zu dem Baum mit erzeugt. Darauf baut das Werkzeug Ag auf ("Attribute Evaluator Generator"), um lokal zugeordnete Regeln zur Berechnung abgeleiteter Werte aus Baumattributen auszuführen.

Die Analyse eines konkreten Files muß als Ergebnis die Unitstruktur bereitstellen. Der Scanner liefert dabei Datenworte, der Parser die Units. Die erkannte Baumstruktur wird so abgelegt, daß ihre Auswertung durch weitere Werkzeuge möglich ist.

5.3. Struktursynthese mittels Estra

Eine Realisierung der Abbildung zwischen Quell- und Zielunits erfolgt durch Erzeugung einer Terminalsequenz der Zielsprache. Ihr liegt ein Strukturbaum der Zielsprache zugrunde, der den Zusammenhang von Zielterminals und Zielnichtterminals beschreibt. Im Gegensatz zum Strukturbaum der Quellsprache wird der Baum der Zielsprache nicht explizit bereitgestellt. Er wird vielmehr durch Belegung des Quellbaums mit Vorschriften ("directives") in Gestalt von Anweisungen ("instructions") einer Programmiersprache implizit dadurch festgelegt, daß bei geordneter Abarbeitung der Anweisungen die gewünschte Zeichensequenz der Zielsprache geliefert wird. Die Semantik ergibt sich damit aus dem Zusammenhang mit den zugehörigen Quellunits über eine solche Generierungsvorschrift. An die Stelle der Objekte zur Festlegung der Bedeutung der Zielmodellunits tritt der Rückgriff auf Quellmodellunits, über deren Trägerobjekte eine indirekte Anbindung an die Objektwelt erfolgt.
Für die Beschreibung der Quellsprache wird noch einmal explizit ein Strukturbaum eingegeben. Die zugrundeliegende, kontextfreie Grammatik ist ein Viertupel G = (Z, P, N, C), wobei Z die Menge der Startsymbole, P die Menge der Produktionen, N die Menge der Nichtterminalen ("Klassen") und C die Menge der Terminalen ("Knoten") bedeutet. Bei den Produktionen werden zwei Typen unterschieden:

- Kettenproduktionen : c0 - c1 (c0, c1 in C)

- Knotenproduktionen : c0 - n (c1, c2, cm) mit n in N und ci in C.

Die Klammerung kann gelegentlich Mißverständnisse hervorrufen, tatsächlich ist sie nur der besseren Lesbarkeit wegen eingeführt ohne sonstige Bedeutung. Bei einer Knotenproduktion können gleichzeitig mehrere Alternativen angegeben werden, die wie üblich durch den jedem Terminal vorangestellten Strich voneinander getrennt werden.
Kernproblem der Überführung ist eine Ermittelung bestehender Zusammenhänge zwischen den Units der Quell- und Zielsprache. Auf der untersten Stufe bedeutet dies je Modellunit

- die Feststellung der zugeordneten Objekte
- die Erfassung der zugehörigen Objekteigenschaften.

Die Quellunits stellen offenbar die Informationslieferanten dar, mit deren Hilfe die im Zielmodell benötigten Attribute bestimmt werden müssen. In einfachen Fällen kann das auf direktem Wege erfolgen. In komplizierten Fällen mit großen Sprachunterschieden empfiehlt sich ein zweistufiges Vorgehen, bei dem als Zwischenschritt die Struktur eines Integrierten Produktdatenmodells einbezogen wird. Dem Modell entspricht eine Sprache, die im Umfeld der universalen Zwischensprache UNCOL zu suchen ist, mit deren Hilfe eine beliebige Quellsprache in eine beliebige Zielsprache durch Bereitstellen nur eines Frontend- und nur eines Backend- Compilers übersetzbar sein soll /AhSU88/. Da auch hier letztendlich eine Unit-Unit-Zuordnung wichtig wird, können wir uns ohne Einschränkung der Allgemeinheit darauf orientieren.
Estra bietet nun unmittelbar die Möglichkeit, Funktionen auf dem Strukturbaum der Quellsprache zu deklarieren, mit deren Hilfe bestimmten zulässigen Mustern von Teilbäumen (" pattern")

- unter möglicher Einbeziehung von Bedingungen und
- unter zusätzlicher Festlegung einer Kostengröße

Instruktionen zugeordnet werden können.
Im Anwendungsfall liegt nach der Analyse des Quelltextes ein konkreter attributierter Syntaxbaum vor. Auf diesen werden entsprechend der allgemeinen Zuordnung von Funktionen zu Teilbäumen die in Frage kommenden Funktionen schrittweise angewendet. Begonnen wird mit dem Aufruf der als erstes vereinbarten Funktion, innerhalb derer jene Muster-Vorschrift-Paarungen aktiviert werden, für die das jeweilige konkrete Muster den Anforderungen des Vergleichsmusters genügt. Dabei können

- eine rekursive Anwendung der Funktion auf und
- der Aufruf weiterer Funktionen für

innere Knoten der Teilbaumstruktur des Musters erfolgen. Bei der Verarbeitung kann direkt auf die aktuellen Attribute des Quell-Strukturbaums durch Angabe des Klassen- und des Attributenamens zurückgegriffen werden, mehrfaches Auftreten derselben Klasse im Muster kann durch zusätzliche Selektoren aufgelöst werden. Zur Erreichung desselben Ziels können dabei unterschiedliche Lösungswege vorbereitet sein, unter denen sich durch eine vorgeschaltete Ermittelung der Kosten bei Bedarf der günstigste auswählen läßt.
Am einfachsten ist offenbar der Einsatz des Werkzeuges Estra zu realisieren, wenn Quellmodellstruktur und Zielmodellstruktur einander ähnlich sind. In diesem Fall sind die in einer Quellunit bereitgestellten Attribute am selben Ort vorhanden, so daß die Zielunitattribute einfach ermittelt und durch Einbau von Schreibaufrufen in die Direktive für eine spätere Ausgabe vorbreitet werden können. Je nachdem, ob die Zielstruktur referenzier-

bare Sätze zuläßt oder nicht, empfiehlt sich für die Codegenerierung eine Vorgehensweise bottom up oder top down. Der Vorstoß gegen ein solches Lokalitätsprinzip bei der Definition von Modellierungssprachen läßt sich nur durch erhöhten Übersetzungsaufwand ausgleichen. Es erfordert zusätzliche Zwischenspeicher oder ein vielfaches Navigieren durch den Strukturbaum des Quellfiles, um die für die aktuelle Zielunit notwendigen Informationen zu beschaffen.
Prinzipiell erfolgt die Konkretisierung der Zielstruktur entsprechend der Ausprägung des konkreten Quellmodells. Über dem Quellmodell wird rekursiv eine Funktion LESEN benötigt, während für das Zielmodell eine Funktion SCHREIBEN wichtig wird. Liegt auch der Stukturbaum des Zielmodells fest, können in jeder Zielunit

- durch Aufruf der LESEN-Funktion für interessierende Teilstrukturen des Quellmodells
- durch Verarbeitung der eingelesenen Attribute zu Ausgabeatributen
- durch Ausgabe der Ausgabeattribute mittels SCHREIBEN-Funktion

die gewünschten Zielzeichensequenzen erzeugt werden.

5.4. Fileanalyse mittels Sprachcompilers

Die Wandlung eines Datenfiles, z. B. eines NIRO-Files, kann unter Verwendung eines Compilers einer höheren Programmiersprache realisiert werden. Dabei wird der enge Zusammenhang zwischen Daten und Verarbeitungsfunktionen bewußt genutzt, wie das in klassischer Form bei der Jackson-Methode des Softwareentwurfs zu Eingabedatenstrukturen methodisches Grundprinzip ist. Dafür gibt es grundsätzlich zwei verschiedenen Möglichkeiten. Der erste Weg besteht darin, die Syntax der Grammatik der Programmiersprache so abzuwandeln, daß typische Konstruktionen eines Datenfiles wie

#20 = Pair(1,2,3)

als Ersatzkonstrukte für Pair(20,1,2,3) oder Pair20(1,2,3) oder Procedure Pair20 akzeptiert werden. Der zweite Weg, auf den hier noch etwas näher eingegangen werden soll, paßt das Eingabefile den Anforderungen der Grammatik an. Das ist realisierbar, weil es sich hier um eine Wandlung lokaler Informationen handelt. Bei Vorhandensein einer entsprechenden Prozedurdeklaration und nach Vorschalten einer einfachen Wandlungsroutine (z.B. auf der Ebene einer Shellprozedur unter UNIX) läßt sich ein korrekter Programmtext erzeugen. Jetzt können die reichen Strukturierungsmöglichkeiten eines Programms unmittelbar genutzt werden, um eine globale Verarbeitung der Attribute von Entities und ihrer vielfältigen Zusammenhänge zu unterstützen. der bei Zugiffsmöglichkeit zu passenden, sprachbezogenen Werkzeugen ein reiches Experimentierfeld eröffnet. So ist es etwa möglich, sofort die Kontrollmechanismen des Parsers zu nutzen, um Syntaxverstöße feststellen zu lassen. Zugleich können damit die operativen Möglichkeiten einer Sprache unmittelbar in der Umgebung der Quelldatei mit genutzt werden. Ein Nachteil zeigt sich offenbar darin, daß wegen der fehlenden Spezifik viele Sprachmöglichkeiten ungenutzt bleiben und damit überflüssigen Ballast bedeuten.
Die Vielfalt programmiersprachlicher Strukturen läßt für ein Datenfile unterschiedliche Gleichsetzungen zu. Eine statische Interpretation geht davon aus, daß jede Entity-Instanz des physischen Files die Vereinbarung einer neuen Prozedur beinhaltet. Typ und Identifikator der Entity-Instanz werden also zu einem Programmnamen verschmolzen, der eine Unterscheidbarkeit der verschiedenen Instanzen gleichen Typs erlaubt. Treten in der Parameterfolge der Instanz Identifikatoren bereits definierter Entity-Instanzen auf, bedeu-

tet das nichts anderes als den Aufruf der zugehörigen Prozeduren. Dazu ist wieder, etwa durch Makrotechnik, der Identifikator durch Hinzunahme der vorbereiteten Typbezeichnung zu vervollständigen.
Da in den Prozeduren die konkreten Attribute von Entity- Exemplaren festgehalten sind, wird auf diese Weise ein Zugang zu den Attributen untergeordneter Entities ermöglicht. Sie können über Parameterliste oder globale, typbezogene Datendeklarationen verfügbar gemacht werden. Eine andere Möglichkeit der Nutzung bieten Schreibanweisungen, die durch Realisierung der Prozeduraufrufe eine zusammenhängende Darstellung aller Attribute eines Entities gestatten. In jedem Fall führen Entity-Exemplare gleichen Typs auf zwei verschiedene Prozedur-Deklarationen, deren Weiterverwendung jedoch mit den Mitteln des Sprachcompilers bzw. Linkers direkt unterstützt wird.
Die dynamische Interpretation faßt von vorneherein jede Entity-Instanz des Eingabefiles als Prozeduraufruf auf, um wieder typabhängig das Wertetupel mit Entity-Identifikator in den Exemplar-Speicher einzutragen. Wird bei der Definition des Entity-Exemplars auf andere Entity-Exemplare zurückgegriffen, erfolgt zur Laufzeit mittels des zugehörigen Entity- Identifikators die Auswahl jener Entity-Instanz, an deren konkreten Attributen Interesse besteht. Die Prozedurvereinbarungen brauchen jetzt pro Typ nur noch einmal hinzugefügt zu werden. Dafür muß bei Rückgriff auf eine andere Instanz der zugehörige Beschreibungssatz zur Laufzeit wieder aufgefunden werden.
Beim Vergleich beider Vorgehensweisen spricht vieles für den dynamischen Ansatz. Er kann perfektioniert werden, indem etwa eine relationale Datenbank zur Speicherung der Entity- Exemplare eines Typs herangezogen wird. Trotzdem sollten die besonderen Chancen der Einzeldeklarationen von Entity-Exemplaren nicht ganz unbeachtet bleiben, erlauben sie doch eine individuelle Ausgestaltung durch Hinzufügen lokaler Bedingungen. Es ist also prinzipiell möglich, hinter jeden Aufruf einer Entity-Instanz Quelltextanweisungen zu stellen, die genau auf die Besonderheiten dieser Instanz Bezug nehmen. Vergleichbares ist im Falle des dynamischen Ansatzes nur dadurch zu erreichen, daß in der Typprozedur alle möglichen Bedingungen gesammelt werden, unter denen dann jedes konkrete Exemplar über Auswahlattribute seine Entscheidung zu treffen hat.
Methodisch bildet die Deklaration jedes Exemplars ein Bindeglied zur Welt der Objekte der objektorientierten Programmierung. So kann durch gemeinsamen Aufruf einer zusätzlichen Klassenprozedur eine Vererbungsmöglichkeit für Attribute und Bedingungen erreicht werden. Werden die Bedingungen so verschärft, daß sie zu funktionellen Abhängigkeiten führen, können ganze Berechnungsabläufe in die Objektwelt hineingetragen werden.
Das Zielfile ist nun durch eine Folge von Ausgaben innerhalb der Entity-Prozeduren zu verwirklichen. Dafür gibt es prinzipiell wieder verschiedene Wege. Am einfachsten würde es sein, wenn die Codeerzeugungsfunktion unmittelbar den Zielcode liefern könnte. Dazu würde der Strukturbaum des konkreten Quellprogramms rekursiv zu durchlaufen sein, wobei der je Knoten bereitstehende Codeanteil durch Schreibanweisungen ausgegeben werden muß. Identifikatoren und Werte, die als Attribute an den Strukturbaum angeheftet sind, sind dabei zu übernehmen und für notwendige Zwischenberechnungen nutzbar zu machen. Dafür ist zu sichern, daß die in den Entity-Prozeduren bereitgestellten Werte bei Aufruf dieser Prozeduren erreicht werden können. In diesem Fall müßten die Prozedurverarbeitungstechniken noch einmal auf der Stufe der Generatorsprache, im Falle von Estra ist das MODULA, nachgebildet werden.
Einfacher zu modellieren, jedoch sicher nicht so effektiv ist ein anderer Weg, bei dem neue Prozeduren erzeugt werden, deren Verarbeitung durch einen Compiler erst die Erzeugung des Zielfiles auslöst. Er bietet jedoch den Vorteil, die Prozedurverarbeitung nicht mehr selbst behandeln zu müssen, die Struktur des Zielfiles klar nachbilden zu können und einen

Compiler einer beliebigen höheren Programmiersprache für anspruchsvollere Aufgaben nutzen zu können.

5.5. Analyse eines NIRO-Demonstrationsfiles

Die Struktur eines NIRO-Files kann durch pUR-Diagramme in einfacher Weise beschrieben werden. Ihren Hintergrund bildet eine Prinzipbeschreibung des Produkts (Bild 20).

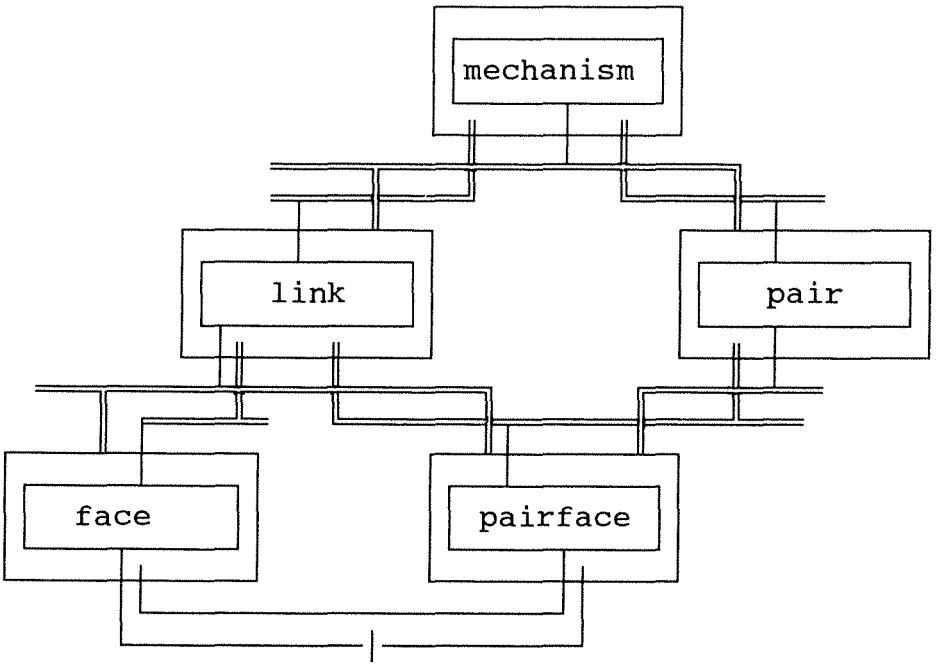


Bild 20. Grundstruktur eines kinematischen Objekts

Sie kann durch zusätzliche Verfeinerung konkreten Anwendungsbedürfnissen angepaßt werden (Bild A1). Für jede Unit ist dabei festgehalten, welche Unterunits sie besitzt und zu welchen Oberunits sie gehört. Für eine schnelle Verarbeitung und eine Sicherstellung von Konsistenz ist eine Verkettung der Units in beiden Richtungen sinnvoll, auch wenn eine vollständige Verkettung in einer der Richtungen oder, wie in NIRO, in einer Mischform bereits ausreicht. Im Falle eines Datenaustausches ist in der Regel ein Neuaufbau im Zielsystem notwendig, so daß bei der Festlegung der Strukturzusammenhänge von vorneherein nur eine der Richtungen benötigt wird (Bild A2).

```
STEP
HEADER
#10=Plac(1,2,3,4,5,6);
#11=Plac(11,12,13,14,15,16);
#12=Plac(21,22,23,24,25,26);
#13=Plac(31,32,33,34,35,36);
#20=Pair(1,2);
#21=Pair(2,3);
#22=Pair(3,4);
#23=Pair(4,5);
#30=PairPlac(#10,#20);
#31=PairPlac(#11,#21);
#32=PairPlac(#12,#22);
#33=PairPlac(#13,#23);
#40=Link((#30,#31));
#41=Link((#31,#32,#33));
#42=Link((#32));
#43=Link((#33));
#50=Joint(#40,#41,#20);
#51=Joint(#41,#42,#21);
#52=Joint(#41,#43,#22);
#60=Structure((#51,#52,#53))
ENDSTEP.
```

Bild 21. Vereinfachtes Muster eines NIRO-Files

Zur Demonstration verwenden wir eine einfache kinematische Musterstruktur, deren Aufbau ein pUR-Diagramm in Bild A3 widerspiegelt. Das zugehörige File (Bild 21) enthält eine kinematische Struktur mit dem Identifikator 60, die aus drei Gliedern Link40, Link41 und Link42 besteht. Es existieren drei Gliedpaarungen Joint50, Joint51 und Joint52, die die angegebenen Links über gemeinsame Koppelstellen (Pairs) verknüpfen.
Zur Durchführung der Strukturanalyse mittels der Werkzeuge Rex, Lalr und Ag greifen wir auf Spezifikationen dieser Werkzeuge zurück, die im Zusammenhang mit einer einfachen Programmiersprache Minilax bereitgestellt wurden. Voraussetzung dafür ist eine Wandlung der Satzstruktur. Da wir uns für eine statische Interpretation entscheiden, muß das File formal in eine Folge von Prozedurdeklarationen umgewandelt werden.

```
echo "
PROCEDURE Joint$1;
DECLARE
C : INTEGER
BEGIN Link$2; Link$3; Pair$4 END Joint$1;"
```

Bild 22. Shellprozedur unter UNIX zur Erzeugung einer Prozedurvereinbarung des Entitytyps Joint

Das gelingt in UNIX einfach durch parametrisierte Shellprozeduren (Bild 22), die einmal für jeden Entity-Typ bereitzustellen sind und durch eine Pipeline-Verkettung der UNIX-Werkzeuge cat, tr, awk und sh die gewünschte Wandlung vornehmen (Bild 23).

```
cat $1 | tr -cs A-Za-z0-9'\012' '\040'|awk '{print "sh " $2 " "$1 " " $3  \
" " $4 " " $5 " " $6 " " $7 " " $8 " " $9 " " }' | sh  in1
```

Bild 23. UNIX-Pipeline zur Umwandlung von Entities in Prozedurvereinbarungen

Das Ergebnis der Transformation für die Joint-Instanzen des Musters enthält Bild 24.

```
PROCEDURE Joint50;
DECLARE
C : INTEGER
BEGIN Link40; Link41; Pair20 END Joint50;
PROCEDURE Joint51;
DECLARE
C : INTEGER
BEGIN Link41; Link42; Pair21 END Joint51;
```

Bild 24. Ergebnis der Verarbeitung des NIRO-Musters durch die Pipeline

6. Schlußbemerkungen

Es ist davon auszugehen, daß die Vernetzung vorhandener, erprobter Teilsysteme ein zentrales CIM-Problem ist, daß auch in nächster Zukunft nichts an Bedeutung einbüßen wird /Scho89/. Damit sind die Vereinheitlichung und Präzisierung von Schnittstellen und eine möglichst rationelle Erzeugung zugehöriger Prozessoren aktuelle Aufgaben von großer Bedeutung.
Parallel dazu wird es wichtig, sich bei der Schaffung neuer Produkte offen für neue Konzepte zu halten und einen methodischen Vorlauf zu sichern. Integration der Fertigung ist nur durch ein gemeinsames Vorgehen von Praktikern und Informatikern zu erreichen. Das setzt eine Modellierungssprache voraus, die auf der untersten Ebene die Produktwelt adäquat beschreibt und von allen Seiten akzeptiert werden kann. Sie sollte langfristig einen Umgang mit Produktdaten gestatten, der dem der Schaffung, Bearbeitung und Nutzung realer Gegenstände nahekommt. Das ist ohne konsensfähige Strukturen und eine saubere Systematik nicht zu erreichen,in die allein eine Modellsemantik eingebettet werden kann. Die letzte GI-Jahrestagung stand unter dem Motto "Die Informatik auf dem Wege zum Anwender". Dieser Weg ist mühsam, da nicht zu jeder Frage eine fertige Antwort existiert. Gerade der Bereich der "Semantik von Sprachen und Daten wird noch nicht einmal annähernd beherrscht" /Seeg90/. In der nüchternen Einschätzung, was die Informatik (noch) nicht geleistet hat und was man von ihr erwarten darf, heißt es bei Seegmüller weiter, daß "wir keine genügend allgemeine Methodik kennen,um

- von Daten zu den darin enthaltenen Informationen zu kommen,
- qualitativ hochwertig automatisch mit natürlichen Sprachen umzugehen, also z.B. Texte zu übersetzen".

Genau auf diesem schwierigen Terrain, auf dem "der Computer noch keine wesentliche und prinzipielle Abstraktionshilfe" ist, hat sich eine Produktdatenmodellierung zu bewähren, die integrierend über die engen Grenzen einer Datensammlung für spezielle Anwendungen hinausgeht. Es bleibt bei Seegmüllers Aufforderung auf der GI-Jahrestagung an die Informatiker: "Leistet den Zeitgenossen , was sie bedürfen, nicht was sie loben".

ANHANG

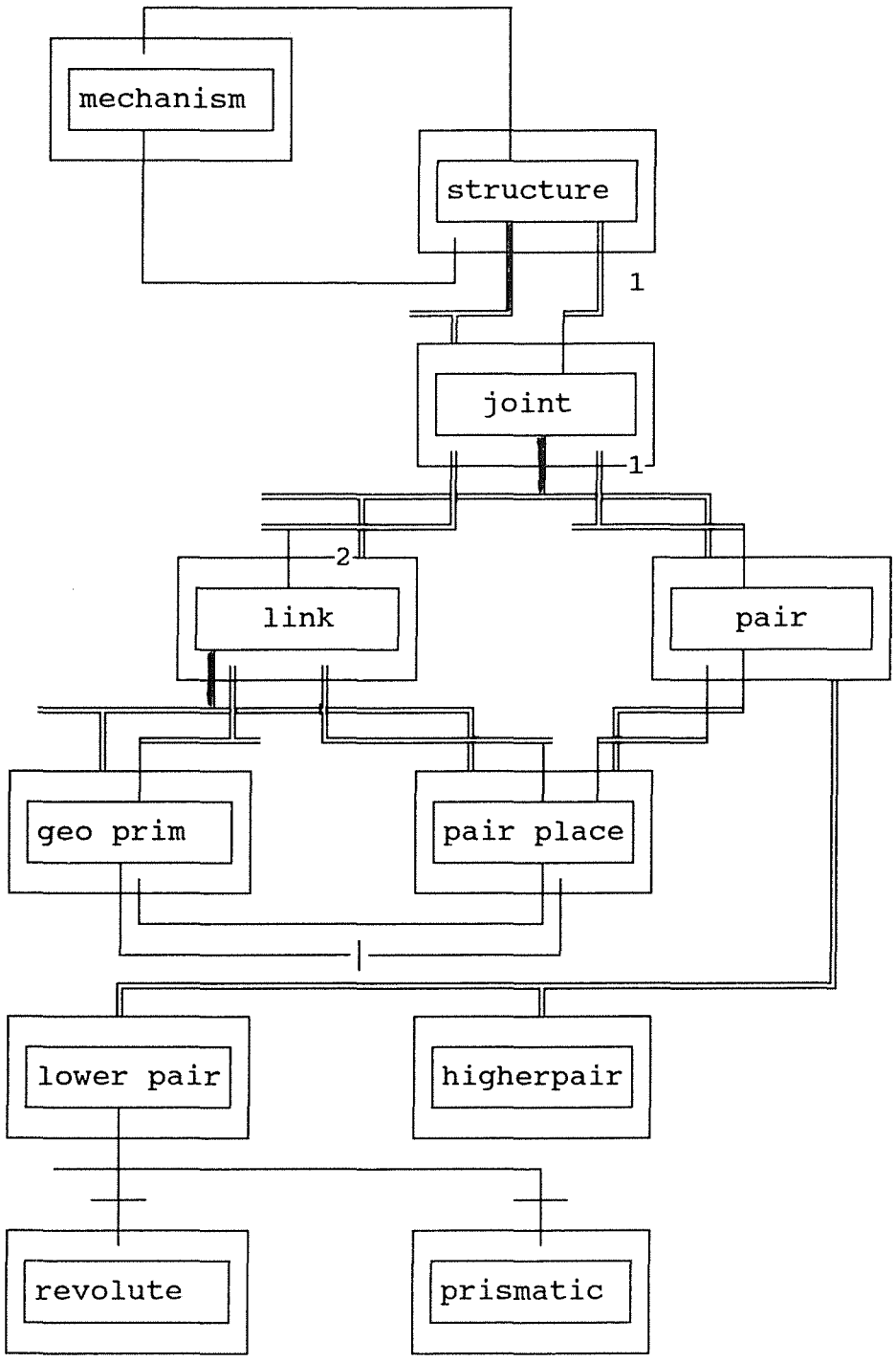


Bild A1. Grundstruktur eines kinematischen Objekts bei NIRO

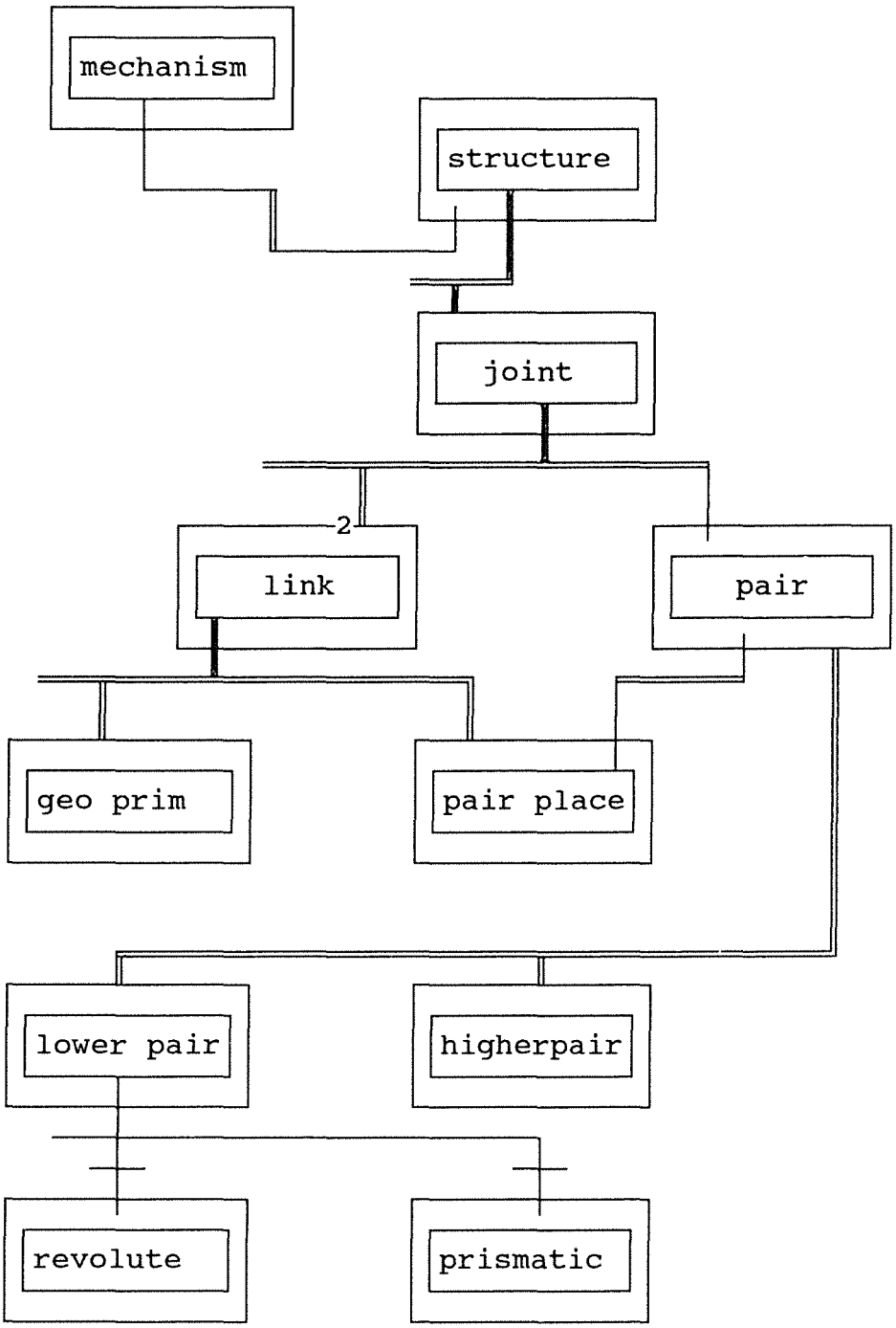
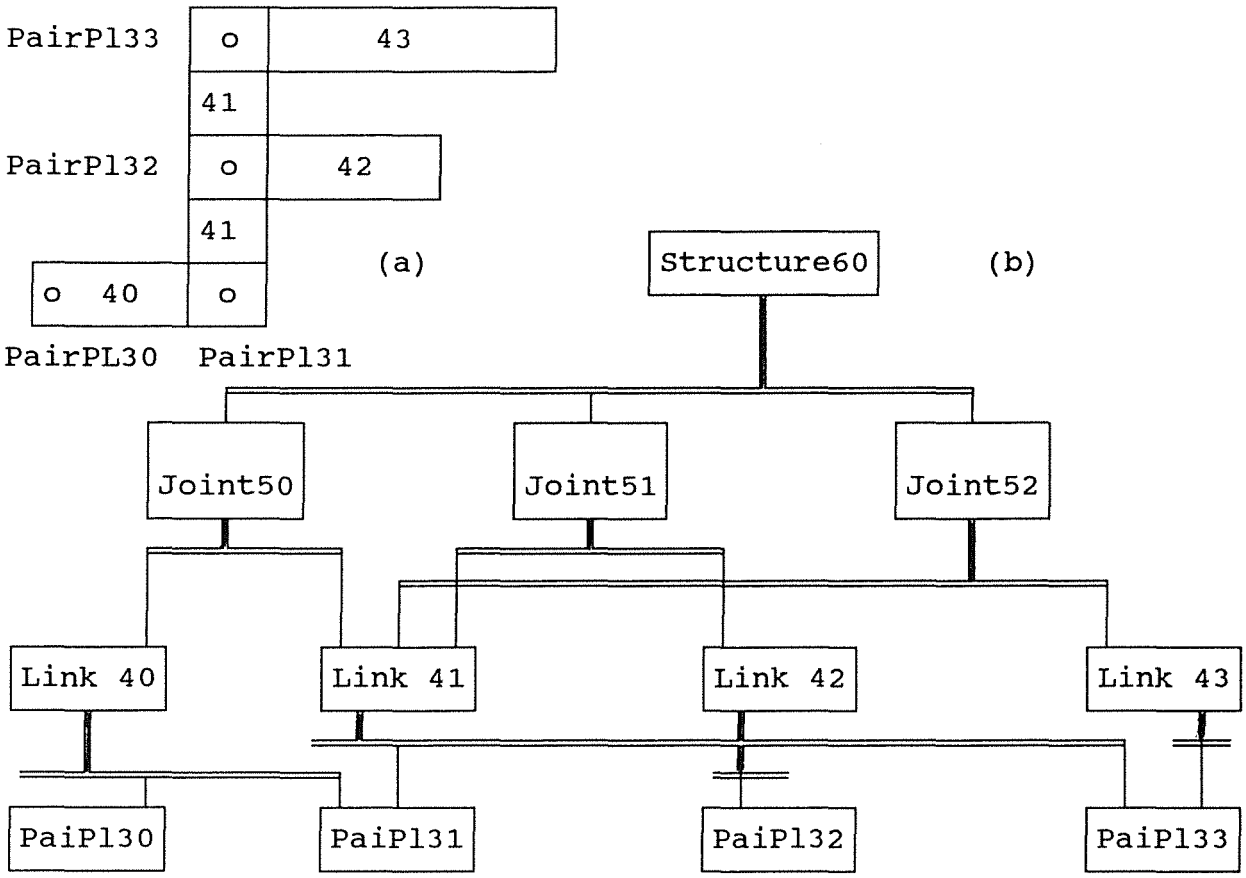


Bild A2. Realisierte NIRO-Struktur



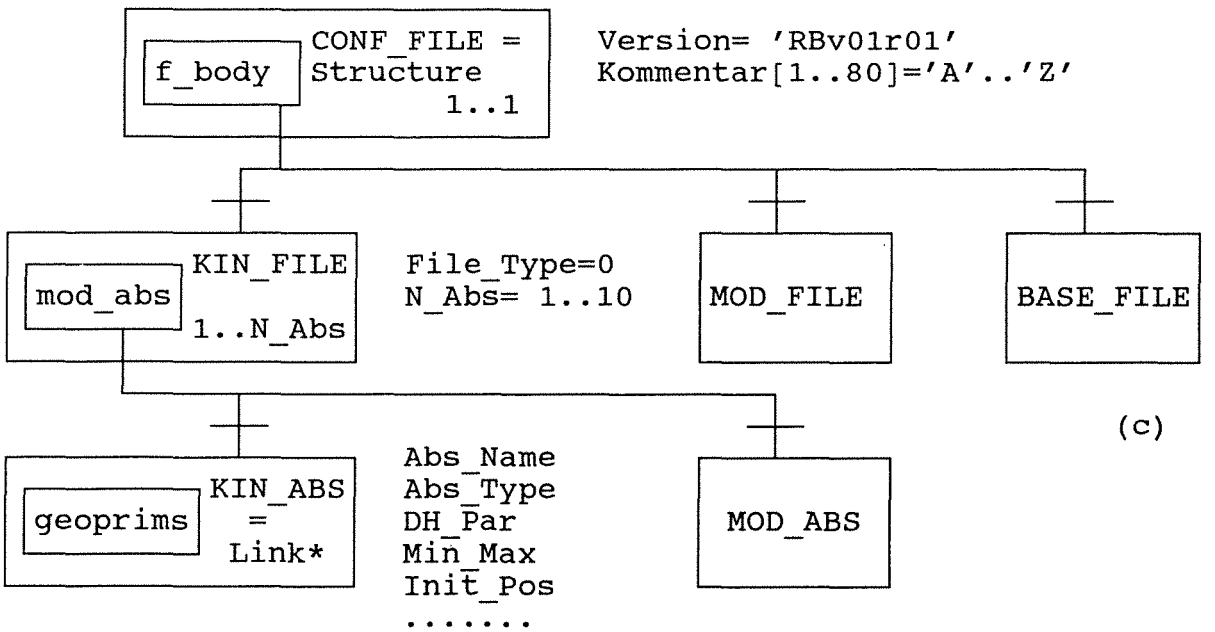


Bild A3. Verschiedene Strukturen des Muster-Mechanismus
a) Physischer Aufbau b)Teilstruktur in NIRO c) Teilstruktur in ROBOT

Literaturhinweise

/AhSU88/ Aho,A.V., Sethi,R., Ullman,J.D.; Compilerbau, Bd.2, Addison-Wesley, 1988.

/Ande89/ Anderl,R.; Integriertes Produktmodell, Zeitschrift für Wirtschaftliche Fertigung,84,1989, S.640-644.

/AnSc87/ Anderl,R.,Schilli,B.; STEP- Eine Schnittstelle zum Austausch integrierter Modelle, ZGDV-Springerbuch,CAD- Schnittstellen in Architektur und Maschinenbau, Springer-Verl.,1987.

/Boos90/ Boos-Bavnbek,B.; Rationalität und Scheinrationalität durch computergestützte mathematische Modellierung,in: Reuter (Hrsg.), 20. GI-Jahrestagung,Stuttgart, Okt. 1990, Proc.I, S.148- 167.

/Chen76/ Chen, P.P.; The Entity- Relationship Model: Towards a Unified View of Data ACM Transactions on Database Systems, Vol.1, No.1, März 1976, S.9-36.

/Danne90/ Danner,W.F.; A Proposed Integration Framework for STEP,Nat. Institute of Stand. and Techn. (NISTIR), Draft, April 1990.

/DaYa90/ Danner,W.F., Yang,Y.; Generic Product Data Model,Version 0.7, NISTIR, Draft, 1990.

/DaYa91/ Danner,W.F., Yang,Y.; Generic Product Data Resources for STEP Version 0.9a, NISTIR, Draft, 2.1.1991.

/DGLo86/ Dittrich,K.R., Gotthard,W., Lockemann,P.C.,DAMOKLES-A Database System for Software Engineering Environments, in: Advanced Programming Environments, Workshop Trondheim,1986,Lecture Notes in Computer Science, Bd. 244, S.353-371.

/DiFe88/ Dijkstra,E.W., Feijen,W.H.J.; A Method of Programming, Addison-Wesley, 1988.

/Dobr89/ Dobrowolny,V.; Über Geometriemodelle mit adaptiven Eigenschaften, Rostocker Informatik-Berichte, Heft 8, 1989.

/Dobr90/ Dobrowolny,V.; Über Beschreibungssprachen für prädikatierte Produktdatenmodelle, in: Richter,D,(Hrsg.), Produktdatenmanagment in der DDR, iir-report der AdW, Heft 4, 1990.

/Grab90/ Grabowski,H. Bedeutung der Normung von Produktmodelldaten in CIM, in: VDI-Berichte 830, Rechnerintegrierte Konstruktion und Produktion, VDI-Verlag 1990,S.75-105.

/GrAS89/ Grabowski,H., Anderl,R, Schmitt,M.; Produktmodellkonzept von STEP, VDI-Zeitschrift, 131, 1989, Nr.12, Seite 84-96.

/Jack87/ Jackson,P.; Expertensysteme, Addison-Wesley,1987.

/KiMc85/ King,R., McLeod,D; Semantic Data Models; in: Bing Yao (Hrsg.), Principles of Database Design, Prentice Hall, 1985, S.115-150.

/KuDo90/ Kupper,H., Dobrowolny,V.; Die Beschreibung von Produktdatenmodellen für den rechnerintegrierten Betrieb mit prädikatierten Unit-Relationship-Modellen, in: Reuter (Hrsg.),20. GI-Jahrestagung, Stuttgart, 1990,Proc.I, S.431-438.

/LeKL91/ Leister,P., Kühnapfel,U., Ludwig,A.; Computer Aided Simulation of a Remote Stream Jet Exchange in a Dissolver Cell, in: Jamshidi,M., Eicker,P.,J. (Hrsg.); Robotics and Remote Systems, Proc. of Fourth ANS Topical Meeting, 1991, S. 353 - 364.

/Gell91/ Gellrich,W.; Compilergeneratoren: Modernes Werkzeug zur Programmierung,in: Chip professional, Programmieren - Aspekte der Computerwissenschaft, Ausgabe 13,1991, S.28-39.

/GrEm90/ Grosch,J., Emmelmann,H.; Project Compiler Construction,GMD, Forschungsstelle an der Uni Karlsruhe, Report 20, 1990.

/Matt89/ Mattos, N.M.; An Approach to Knowledge Base Managment, Diss. Uni Kaiserslautern, Fachbereich Informatik, 1989.

/Refe89/ Referenzmodell für CAD- Systeme; Gesellsch.für Informatik, Fachgr. 4.2.1, 1989.

/Schl88/ Schlechtendahl,E.G., Specification of a CAD*I Neutral File for CAD, Geom. Version 3.3, Springer Verlag,1988.

/SchW90/ Schlechtendahl,E.G., Weick,W.; Esprit Contributions to the Exchange of CAD Models, in: CADCAM in Europe, Official Yearbook 1990, S.15-24.

/Seeg90/ Seegmüller,G.; Informatik auf dem Wege zum Anwender, Informatik-Spektrum 1990, 13, S.307-310.

/Viel89/ Vielsack,B.; Spezifikation und Implementierung der Transformation attributierter Bäume, Diplomarbeit, GMD Forschungsstelle an der Uni Karlsruhe, 1989.

/Will89/ Williams,A.L.; Comparative Analysis of the Modeling Languages: IDEF1X, EXPRESS, NIAM, NIST, ISO TC 184 / SC4 Secretariat, Gaithersburg, MD, USA.

/WaPI91/ Wasseman,A.I., Pircher,P.A.; Objekte und Struktur in C + + in: Chip professional,Programmieren - Aspekte der Computerwissenschaft, Ausgabe 13,1991, S.28-39.