10. SYMPOSIUM "FERTIGUNGSGERECHTES KONSTRUIEREN"
SCHNAITTACH, 14. UND 15. OKTOBER 1999

IMPLIZITE UND EXPLIZITE BESTANDTEILE DES PRODUKTMODELLS UND IHRE BEDEUTUNG FÜR DIE ENTWICKLUNG VON CAX-SYSTEMEN

C. Weber, H. Werner

Kurzfassung

Softwareunternehmen und Forschnugsinstitute arbeiten an der Integration neuer Funktionalitäten in CAx-Systeme, insbesondere der Verarbeitung nicht-geometrischer Eigenschaften. Der derzeit übliche Weg besteht darin, die Semantik dieser Eigenschaften im Cax-System fest zu „verdrahten". Dies führt dazu, daß wesentliche Bestandteile des Produktmodells implizit im System enthalten sind und damit anderen Systemen nicht zugänglich sind. Daher können die mit großem personellen Aufwand an verschiedenen Instituten erarbeiteten Lösungen für einzelne schmale Teilbereiche nicht zu einem universellen Gesamtsystem zusammengeführt werden. Das Bereitstellen eines solchen Gesamtsystems aus *einer* Hand ist jedoch aufgrund des immensen Entwicklungsaufwandes nicht möglich. Einen Ausweg bietet die vollständige Verlagerung der neuen Funktionalitäten in das explizite Produktmodell mit Hilfe einer objektorientierten Beschreibungssprache. Eine solche Sprache könnte gemeinsam von allen auf diesem Gebiet arbeitenden Instituten und eventuell Systementwicklern spezifiziert werden. Die Implementierung eines Basissystems, das diese Sprache interpretiert, erfordert einen vergleichsweise geringen Aufwand und macht den Weg frei für eine Bündelung der Kräfte im Bereich der rechnerunterstützten (insbesondere Feature-basierten) Konstruktion.

1 Produktmodelle in konventionellen CAx-Systemen

Grundlage des (rechnerunterstützten) Konstruktionsprozesses in der Praxis sind die breit eingeführten CAD-Systeme. Die Abbildung der geometrischen Elemente im Produktmodell findet sich jedoch streng genommen nur zum Teil in den vom CAD-System explizit erzeugten Daten(-strukturen); ein anderer Teil ist (implizit) im erzeugenden bzw. verarbeitenden System „verdrahtet" (**Abb. 1, „konventionelles System"**).

So würde man z.B. zu einer Kugelmantelfläche Mittelpunkt und Radius im Datenmodell finden. Diese Größen haben jedoch ohne die im System hinterlegten Zusatzinformationen, z.B. darüber, wie die Fläche dargestellt und mit anderen Geometrieelementen verknüpft werden kann, keinerlei Bedeutung.

Schnittstellenstandards wie IGES, VDA-FS oder auch STEP sind letztlich Maßnahmen, die nicht nur die expliziten, sondern gerade auch die impliziten Bestandteile des Produktmodells vereinheitlichen und allgemein zugänglich machen sollen. Sie geben an, welches Wissen in ein System hineincodiert werden muß, damit es die expliziten Bestandteile des Produktmodells richtig interpretieren kann. Dieser festcodierte, implizite Anteil spiegelt das Bild, das man von dem Produkt hat, oder wie Andreasen präziser sagt, das „phenomenon model" [Jens-97] wieder. Das „phenomenon model" für Geometrieelemente ist in der Mathematik seit langer Zeit festgelegt und kann daher ohne Bedenken fest codiert werden.

Um das CAx-System mit mehr „Wissen" (über Produkte und/oder den ggf. unternehmensspezifischen Konstruktionsprozeß) auzustatten, ist seit langer das Werkzeug der Makro-/Variantenprogrammierung bekannt. In jüngster Zeit wird der Zugang dazu durch parametrische Systeme erleichtert, die dann bereits das Prädikat „Feature-basiert" beanspruchen. Alle die-

se Lösungen bleiben jedoch nach wie vor im Bereich der Geometrie verhaftet (**Abb. 1, „Features A“**). Der Austausch von in Makros, Variantenprogrammen, parametrisierten Geometriemodellen und Form-Features enthaltenen Informationen ist zwar grundsätzlich möglich (siehe z.B. VDA-PS), bereitet in der Praxis aber bis heute große Probleme.

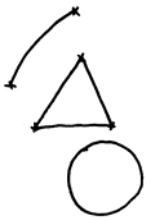
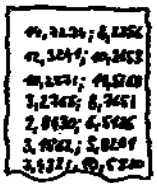
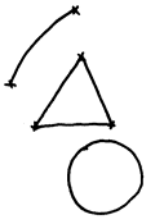
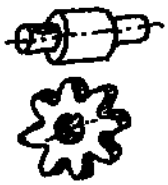
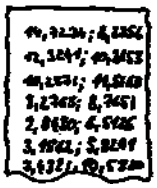
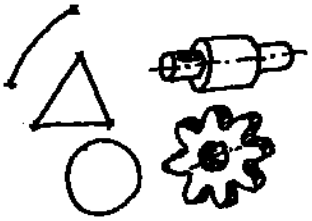
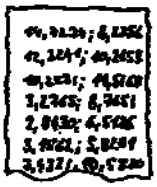
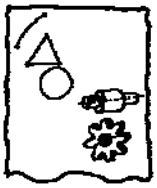
Systemklasse	**Implizite Bestandteile**		**Explizite**	**Wissens-**	**Austausch-**
	im System	**in Zusatzmodulen**	**Bestandteile**	**gehalt**	**barkeit**
Konventionell				-	+
„Features A“: • konventionell mit Makros u. Varianten • parametrisch • Form-Features				o	o
„Features B“: Featurebasierte Systeme in der Forschung				+	-
„Features C“: Beschreibungs-sprache mit Interpreter(n)				+	+

Abb. 1: *Modellkonzepte in CAD-Systemen*

2 Feature-basierte Systeme in der Forschung

Bei nichtgeometrischen Eigenschaften stellen sich neue Probleme: Schon bei einem alltäglichen Element wie dem Werkstoff ist es schwer zu entscheiden, welche Merkmale in einem CAx-System abgebildet werden sollten. Für gewöhnliche FEM-Rechnungen genügen die Streckgrenze, der E-Modul, die Querkontraktionszahl und evtl. die Dichte. Unter Umständen müssen aber auch Fließ- und Kriechkurven berücksichtigt werden. Auch die Wärmedehnung ist für viele Berechnungen von goßer Wichtigkeit. Nicht vergessen werden sollten Ermüdungsverhalten, Kerbschlagarbeit, Korrosionsverhalten, elektrochemisches Potential und Gefüge.

Die Bedeutung all dieser Merkmale müßte nach heutigem Stand der Technik in einem CAx-System hinterlegt werden, damit es ein entsprechend standardisiertes Produktmodell lesen und verarbeiten kann. Es wird deutlich, daß der Implementierungsaufwand immens ist, obwohl die meisten Merkmale eher selten benötigt werden. Dabei ist noch nicht einmal garantiert, daß wirklich alle Anwendungsfälle abgedeckt sind. Aus diesem Blickwinkel ist es verständlich, daß sich kommerzielle Systementwickler erst gar nicht an eine echte Verarbeitung nicht-geometrischer Eigenschaften heranwagen und auch STEP kaum zu mehr als reiner Geometriedatenübertragung genutzt wird.

Im Bereich der universitären Forschung wird jedoch seit geraumer Zeit an Systemen gearbeitet, die den Konstruktionsprozeß wesentlich umfassender als herkömmliche CAD-Systeme unterstützen können - die Schlagworte sind hier „Feature-basiert“ oder sogar „wissensbasiert“. Die Prototypen werden üblicherweise mit hohem personellen Aufwand als monolithische Systeme realisiert und über Programmierschnittstellen an kommerziell erhältliche CAD-Systeme angekoppelt. Sie decken meist nur eine schmale Nische im Bereich der potentiellen Konstruktionsunterstützung ab, lassen sich aber mangels eines geeigneten Produktmodells bzw. Datenaustauschformates nicht miteinander kombinieren (**Abb. 1, „Features B“**).

Besonders schwierig ist die Situation bei der Unterstützung der frühen Phasen des Konstruktionsprozesses, da hier i.d.R. noch gar keine Geometrie existiert und der gerne gewählte (Aus-)Weg, Features an Geometrieelemente zu knüpfen, nicht offensteht.

Man könnte die derzeitige Situation zusammenfassend dadurch charakterisieren, daß die zunehmende Integration von Konstruktionswissen in CAx-Systeme zwangsläufig eine Spezialisierung erfordert, wodurch ein besonderer Bedarf der Kombinierbarkeit von Einzellösungen entsteht. Gleichzeitig führt aber die steigende Spezialisierung und die damit einhergehende Verfeinerung des impliziten Produktmodells („phenomenon model“) zu einer immer geringeren Austauschbarkeit von Daten gegenüber reinen Geometriemodellen.

3 Vollständig explizites Produktmodell

Den Ausweg aus dem Dilemma kann nur ein Produktmodell bieten, das gleichzeitig eine hohe Spezialisierung erlaubt und dennoch die vollständige Übertragbarkeit von Produktdaten ohne übertriebenen Implementierungsaufwand gewährleistet.

Dies erfordert aber, daß das gesamte Produktmodell explizit gemacht wird, d.h. daß das „phenomenon model“ nicht mehr fest in das CAx-System codiert, sondern nur bei Bedarf geladen wird. Dazu muß seine Beschreibung im Datenaustauschformat enthalten sein. Da die Verhaltensbeschreibung einen wesentlichen Anteil des „phenomenon model“ bildet, kommt hierfür nach heutigem Stand der Technik nur eine objektorientierte Beschreibungssprache in Frage (**Abb. 1, „Features C“**).

Die in dieser Sprache formulierten Objekte können einfache Geometrieelemente, aber auch Features oder „design units“ im Sinne von Mortensen [Mort-97] repräsentieren. Sie enthalten Attribute (Daten) und Verhaltensbeschreibungen (Methoden). Damit die einzelnen Objekte sinnvoll miteinander kombiniert werden können, sollten sie über Schnittstellen verfügen, die beispielsweise Wirkflächen repräsentieren. Die Abbildung von Aggregations- und Abstraktionshierarchien ist ebenfalls erforderlich.

Am Lehrstuhl für Konstruktionstechnik/CAD in Saarbrücken ist eine solche Beschreibungssprache und das zugehörige Modelliersysten „Ligo“ entstanden [WeWe-98a, WeWe-98b]. Obwohl Ligo außer Drahtgittermodellen zur Visualisierung keine expliziten Geometriemodelle

verarbeitet, bietet es bereits ein breites Anwendungsspektrum. In Ligo modellierte Objekte können verwendet werden, um

- Dimensionierungsrechengänge durchzuführen und Festigkeitsnachweise zu erstellen,
- Simulationen und Überschlagsrechnungen in frühen Phasen des Konstruktionsprozesses durchzuführen,
- Anhand von Randbedingungen geeignete Normteile aus Datenbanken auszuwählen,
- Nicht formalisierbares Konstruktionswissen zu Lösungselementen in Form angehängter Dokumente zur Verfügung zu stellen,
- Lösungsalternativen parallel zu verwalten und automatisch Bewertungsmatrizen zu erstellen/auszuwerten und
- automatisch Modelle für andere CAx-Programme zu erzeugen.

Ein Konstruktionsobjekt in der Beschreibungssprache von Ligo sieht etwa so aus:

```
Konstruktionsobjekt Pleuel: geometrisch
Text: schematisches Modell einer Pleuelstange

Schnittstellen:
Nabensitz außen Pleuelauge_1
Nabensitz außen Pleuelauge_2

Parameter:
skalar l,Achsabstand,[mm]
skalar t,Dicke,[mm]
Darstellung Gittermodell: 3D

Methoden:
intern "Neu": folgt
  frage .l
  frage .t

  Positioniere Pleuelauge_1:v(0;0;0)[mm]; Einheitsmatrix
  Positioniere Pleuelauge_2:v(0;.l;0)[mm]; Einheitsmatrix

  Darstellung Gittermodell:

  ... (Befehle zur Geometrieerzeugung ausgelassen) ...

  /Darstellung
/ende

intern "Betriebszustand": folgt
  'Summe aller Kräfte=0, Zwangskraft zurückspeisen
  .Pleuelauge_1.F=v(0;
  -(.Pleuelauge_2.F:x^2/.Pleuelauge_2.F:y + .Pleuelauge_2.F:y);0)[N]
  .Pleuelauge_2.F= -(.Pleuelauge_1.F)
/ende

rechengang "Festigkeitsrechnung": folgt
  Parameter:
  skalar h_m,"Mittlere Höhe",[mm]
  skalar p,"Flächenpressung",[N/mm^2]
  skalar A,"Fläche",[mm^2]
  skalar I,"Flächenträgheitsmoment",[mm^4]
  skalar I',"Trägheitsradius",[mm]
  skalar \l,"Schlankheitsgrad"
```

```
  skalar \s_k,"vorhandene Spannung",[N/mm^2]
  skalar \s_z,"Kritische Spannung",[N/mm^2]
  skalar F,"Längskraft",[N]

  .F=.Pleuelauge_2.F:y

  \bf Flächenpressung im oberen Pleuelauge: \/bf
  .A=.Pleuelauge_2.d_i * .t
  .p=.F/.A

  \bf Flächenpressung im unteren Pleuelauge: \/bf
  .A=.Pleuelauge_1.d_i * .t
  .p=.F/.A

  \bf Knickung des Pleuels: \/bf
  .h_m=.Pleuelauge_1.d_i + .Pleuelauge_2.d_i / 2
  .A=.h_m * .t
  .I=.t^3 * .h_m / 12
  .I'=sqrt(.I/.A)
  .\l=.l/.I'
  .\s_z=Pi^2 * s(210000)[N/mm^2]/ .\l^2
  .\s_k=.F/.A

/ende
```

Der Ersteller von Konstruktionsobjekten arbeitet allerdings nicht mit dieser Beschreibungssprache, sondern mit einem Editor, der die Eigenschaften und Methoden übersichtlich darstellt.

In [WeWe-98b] wurde auch die Implementierung des Chromosomen-Modells mit Hilfe von Ligo-Objekten beschrieben. Interessant ist dabei, daß es mit einem expliziten Produktmodell grundsätzlich möglich ist, verschiedene „phenomenon models“ zu mischen, also z.B. ein auf Basis der „domain theory“ [Andr-80] erstelltes Modell eines Maschinenelementes in einer auf konventionellere Weise dargestellten Baugruppe zu verwenden und umgekehrt.

4 Definition einer neuen Beschreibungssprache

Wenn alle Institute, die Forschung im Bereich der rechnerunterstützten Konstruktion betreiben, ihr jeweiliges Produktmodell auf Basis einer allgemein akzeptierten Beschreibungssprache explizit formulieren würden, ließen sich die an verschiedenen Orten erarbeiteten (Teil-) Modelle und Funktionalitäten in *einem* System nutzen und miteinander kombinieren. Dies würde auch einen wesentlichen Impuls zur Übernahme von neuen Ansätzen in kommerzielle Systeme sowie zur Weiter-/Neuentwicklung von Schnittstellenstandards geben.

Dazu müßte „lediglich“ eine solche Sprache zur Beschreibung von Produktmodellen und deren Verhalten spezifiziert werden (nicht das Produktmodell an sich!), was im Idealfall in Zusammenarbeit zwischen allen interessierten Instituten geschieht. Sie ist grundsätzlich nichts anderes als eine problemangepaßte objektorientierte Programmiersprache für den Bereich der Konstruktion, die von verschiedenen miteinander konkurrierenden Systemen interpretiert werden kann. Eine solche Sprache wurde z.B. im Bereich der Simulation dynamischer Systeme bereits spezifiziert (MODELICA). Die mit der Entwicklung von Ligo für ein größeres Anwendungsgebiet gemachten Erfahrungen zeigen, daß der erforderliche Sprachumfang überschaubar ist und demzufolge auch die Implementierung vergleichsweise wenig Aufwand erfordert.

Die gesamten Funktionalitäten existierender Feature-basierter Systeme können dann in dieser Sprache formuliert werden und stehen damit zur allgemeinen Verwendung zur Verfügung. Dazu wird (mindestens) ein Basissystem bzw. eine Entwicklungs- und Laufzeitumgebung benötigt, welche die Sprache interpretiert, selbst aber keine eigene „Intelligenz“ mehr benötigt (diese ist ja ins Produktmodell verlagert). Dieses Basissystem muß lediglich folgende Grundfunktionalitäten bereitstellen:

- vektorielles und skalares Rechnen mit einheitenbehafteten phys. Größen
- Koordinatentransformationen
- Handhabung von Schnittstellen zwischen den Bausteinen
- Handhabung von Mengen
- elementare Dateioperationen
- Geometrieerzeugung und -manipulation
- Dokumentation analytischer Berechnungen inkl. Formelsatz
- Aufrufen externer Programme
- Verzweigung, Schleifen
- Erzeugung und Manipulation von Relationen
- Constraint-Verarbeitung
- Verwaltung verschiedener „Wissensmodule“, die einander bedingen können

Darüber hinaus muß es eine Benutzerschnittstelle mit folgenden Funktionen besitzen:

- Editor für Feature-Struktur (inkl. Verknüpfen von Schnittstellen)
- Erzeugen/Löschen von Features
- Rückfragen der Methoden an den Benutzer ermöglichen
- Spezialisieren von Features
- Manipulation der Attribute
- Ausführen von Methoden
- Navigation im Netz der Relationen inkl. Filter und Suchfunktionen
- Visualisierung von 2D/3D-Geometrie
- Anzeigen von Dokumenten

5 Literaturverzeichnis

[Andr-80] M.M. Andreasen: Machine Design Methods Based on a Systematic Approach – Contribution to a Design Theory. Diss. Lund Institute of Technology 1980.

[Jens-97] T. Jensen: An Empirical Study of Variant Design With a Designer's Workbench. Proceedings of ICED 97, Tampere, WDK 25, Vol. 3, pp. 277-282

[Mort-97] N.H. Mortensen: Design Caracteristics as Basis for Design Languages. Proceedings of ICED 97, Tampere, WDK 25, Vol. 2, pp. 23-30, Heurista, 1997.

[WeWe-98a] H. Werner, C. Weber: LIGO – an Object-Oriented Modelling Tool for IPD. 2nd International Workshop „Integrated Product Development“ (IPD 98), Magdeburg 17.-18.09. 1998. Tagungsband S. 44-51. Otto-von-Guericke-Universität, Magdeburg 1998.

[WeWe-98b] H. Werner, C. Weber: Eine Implementierung des Chromosomenmodells mit Hilfe des objektorientierten Konstruktionssystems Ligo. 9. Symposium „Fertigungsgerechtes Konstruieren", Schnaittach/Erlangen 15.-16.10.1998. Tagungsband S. 143-148.

Prof. Dr.-Ing. C. Weber, Dipl.-Ing. H. Werner
Universität Saarbrücken, Lehrstuhl für Konstruktionstechnik/CAD
Postfach 15 11 50, 66041 Sarbrücken
Tel: 0681-302 3075, Fax: 0681-302 4858
e-mail: weber@cad.uni-sb.de, werner@cad.uni-sb.de