9. SYMPOSIUM "FERTIGUNGSGERECHTES KONSTRUIEREN"
SCHNAITTACH, 15. UND 16. OKTOBER 1998

EINE IMPLEMENTIERUNG DES CHROMOSOMENMODELLS MIT HILFE DES OBJEKTORIENTIERTEN KONSTRUKTIONSSYSTEMS LIGO

H. Werner, C. Weber

Zusammenfassung

Ein grundlegendes Problem in der rechnerunterstützten Konstruktion ist das Fehlen eines zufriedenstellenden Produktmodells. Durch Verwendung eines objektorientierten Informationsmodells läßt sich jedoch ein Konstruktionssystem implementieren, das die gleichzeitige Verwendung verschiedener (problemangepaßter) Produktmodelle erlaubt. Das Chromosomenmodell eignet sich als Basis solcher Modelle, da es ihnen eine gewisse Struktur vorgibt und dennoch genügend Raum für die fallspezifische Anpassung läßt.

Das am Lehrstuhl für Konstruktionstechnik/CAD in Saarbrücken entwickelte Modelliersystem „Ligo" erlaubt den praktischen Einsatz eines solchen Konzepts. Es verwaltet Konstruktionselemente als gekapselte Bausteine, die ihre eigene Verhaltensbeschreibung enthalten und über definierte Schnittstellen miteinander kombiniert werden können.

1 Aktuelle Defizite bei der Produktmodellierung

Zur Unterstützung (von Teilen) des Konstruktionsprozesses existiert inzwischen eine unüberschaubare Vielzahl von einzelnen Programmen. Jedoch sind in der aktuellen Forschung und Softwareentwicklung drei wichtige Entwicklungsrichtungen zu beobachten:

- Die Integration verschiedener Systeme,
- das Modellieren von Bauteilfunktionen,
- die Unterstützung der frühen Phasen des Konstruktionsprozesses.

Dabei krankt die Umsetzung vorhandener Konzepte in Anwendersoftware vor allem am Fehlen eines universell einsetzbaren Produktmodells, das alle interessierenden Eigenschaften und Zusammenhänge enthält, alle Phasen des Konstruktionsprozesses abdeckt und für eine automatische Informationsverarbeitung geeignet ist. Ein festgeschriebenes konkretes Produktmodell kann die notwendige Universalität grundsätzlich nicht erreichen.

Einen flexibleren Ansatz stellt dasChromosomenmodell [Ferreirinha 90] dar: Genau genommen ist es eine Klasse von Modellen, da es weder die zu modellierenden Eigenschaften einzeln benennt, noch informationstechnische Details festlegt, sondern nur die Modellstruktur. Dadurch bleibt es einerseits universell, erfordert aber andererseits eine Interpretation, um umgesetzt werden zu können. Das bedeutet für ein konstruktionsunterstützendes System, daß es das Produktmodell im Detail nicht von vornherein kennen kann.

2 Objektorientierte Modellierung in Ligo

Einen Ausweg bietet die objektorientierte Modellierung: Die Tools werden einfach als Methoden in das Modell integriert. Dadurch wird sichergestellt, daß für jedes Tool auch die

benötigten Eigenschaften im Modell enthalten sind. Ein nachträgliches Detaillieren oder Hinzufügen von Funktionalität ist durch (Mehrfach-) vererbung möglich.

Ligo (lat. "ich verbinde") ist als Basis für ein Bausteinsystem konzipiert. Die einzelnen Bausteine können dabei Baugruppen, Einzelteile, Querschnitte, Werkstoffe, Rechengänge oder sogar Prozesse repräsentieren.

Wenn ausreichend viele elementare Bausteine in einer Bibliothek zur Verfügung stehen, reduziert sich der Modellierungsprozeß weitgehend auf die Kombination und Spezialisierung von Bausteinen. Bausteine, die nicht in der Bibliothek vorhanden sind, können mit Hilfe eines Editors schnell erstellt und in das Gesamtmodell eingefügt werden. Die Bausteine in Ligo sind „Konstruktionsobjekte“.

2.1 Definition des Begriffes „Konstruktionsobjekt“

Ein Konstruktionsobjekt ist ein *abgeschlossenes* Informationsmodell, das einen Teil eines technischen Systems beschreibt und sich im Sinne eines Bausteins mit anderen Konstruktionsobjekten zur Beschreibung komplexer Systeme kombinieren läßt.

„Abgeschlossen“ bedeutet, daß sich ein Konstruktionsobjekt auch einzeln und unabhängig von anderen Konstruktionsobjekten verwenden läßt.

Es hat folgende Eigenschaften:

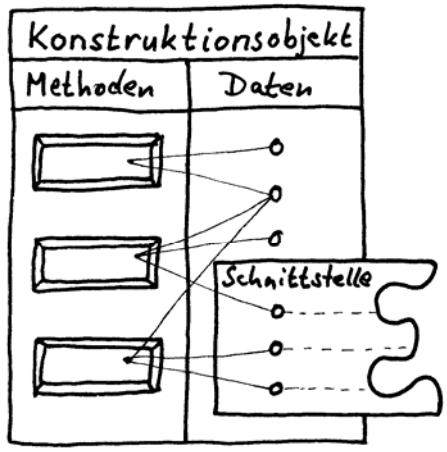

Abb. 1: Schema eines Konstruktionsobjekts

- Es enthält *Daten* (Skalar, Vektor, Matrix, Text oder Verweis auf Datei)
- Es enthält eine Verhaltensbeschreibung in Form von *Methoden*, die die Daten manipulieren können.
- Es enthält *Schnittstellen*, die Daten nach außen hin verknüpfen.
- Es kann Konstruktionsobjekte als Teile enthalten und hat dann Zugriff auf deren Methoden und Daten.
- Es kann ein eigenes lokales Koordinatensystem besitzen
- Es kann durch *Ableiten* (Spezialisierung) zusätzliche Eigenschaften und Methoden erhalten und durch Wegnahme von Eigenschaften abstrahiert werden.

Generell muß zwischen der Klasse und der Instanz eines Konstruktionsobjektes unterschieden werden: Die Klasse enthält nur die Struktur und die Methoden, während eine Instanz die individuellen Daten enthält. Im folgenden wird mit Konstruktionsobjekt die Instanz bezeichnet. Die Methoden in Ligo können entweder in einer sehr einfachen Sprache formuliert werden, die vom System interpretiert wird, oder als externe Programmaufrufe realisiert werden. Dadurch kann die Funktionalität existierender Software genutzt werden.

2.2 Definition des Begriffes „Schnittstellenobjekt“

Ein Schnittstellenobjekt besteht aus zwei Seiten, die jeweils einen Satz von Daten enthalten. Zwischen beiden Seiten ist eine eineindeutige Zuordnung (Bijektion) definiert. Eine Schnittstellenseite, die Bestandteil eines Konstruktionsobjekts ist, kann mit dem zugehörigen

Gegenstück in einem anderen Konstruktionsobjekt verbunden werden. Dadurch werden die einander zugeordneten Daten automatisch synchronisiert. Gegebenenfalls wird dabei eine automatische Koordinatentransformation vorgenommen, so daß die verbundenen Konstruktionsobjekte selbst nichts über ihre relative Lage zueinander wissen müssen.

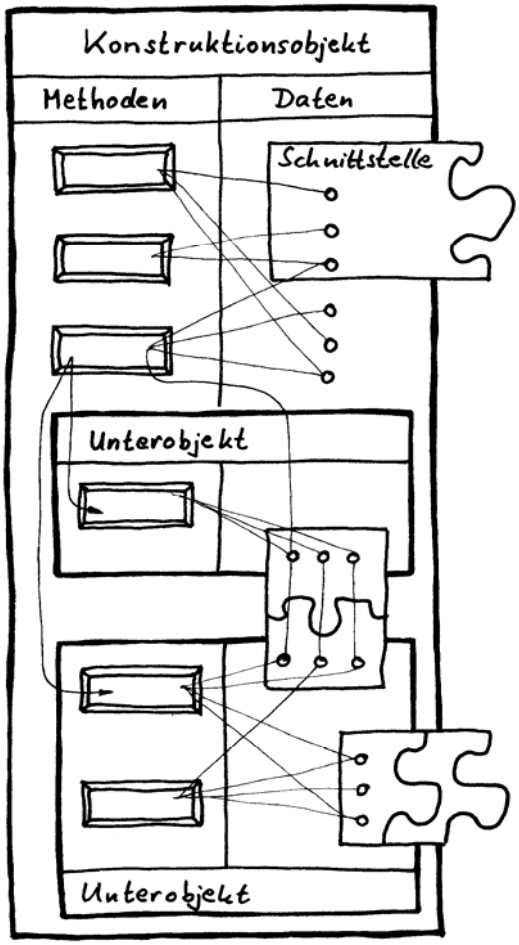

Abb. 2: Zusammengesetztes Konstruktionsobjekt

2.3 Relationen zwischen Konstruktionsobjekten

So wie zwischen den Schnittstellen Relationen der Art „ist verknüpft mit“ existieren, gibt es auch eine Reihe von Relationen zwischen Konstruktionsobjekten, die teilweise mit Mechanismen verbunden sind. Die wichtigsten sind:

- „enthält“ und „ist Bestandteil von“: Abb. 2 zeigt ein Konstruktionsobjekt, das zwei Unterobjekte enthält. Das Oberobjekt kann alle Daten und Methoden der Unterobjekte nutzen.
- „ist Verallgemeinerung von“ und „ist Spezialfall von“. Bei der Definition von Spezialfällen wird der aus der objektorientierten Programmierung bekannte Vererbungsmechanismus genutzt.

2.4 „Plug-Ins“

Das Konzept der Plug-Ins in Ligo ist eine Verallgemeinerung der Mehrfachvererbung. Ein Plug-In fügt Daten, Methoden und Schnittstellen zu einem existierenden Objekt hinzu. Es kann jedoch die Anwesenheit anderer Plug-Ins oder Basisklassen voraussetzen und auf deren Daten und Methoden zugreifen. Ein Plug-In, das keine solchen Voraussetzungen macht, ist eine Basisklasse. Plug-Ins ermöglichen eine nachträgliche Abstraktion von Konstruktionsobjekten.

2.5 Weitere Objekte und Relationen

Neben Konstruktionsobjekten, Schnittstellenobjekten und Plug-Ins lassen sich in Ligo Personen, Dateien und Begriffe eintragen. Zwischen all diesen Objekten lassen sich bestimmte Relationen knüpfen. Mit Hilfe der Relationen kann ein semantisches Netz zur Strukturierung der Information aufgebaut werden. In diesem kann man sich entweder von Objekt zu Objekt bewegen oder mit Hilfe von Filtern nach bestimmten Objekten suchen. Auch die Methoden der Konstruktionsobjekte haben Zugriff auf das semantische Netz.

3 Einbettung des Chromosomenmodells in Ligo

Der Begriff Chromosomenmodell geht auf [Ferreirinha 90] zurück. Nach [Mortensen, Andreasen 96] beschränkt sich das Chromosomenmodell auf statische (‚structural‘)

Eigenschaften, von denen Modelle für Funktion und Verhalten abgeleitet werden. Da Ligo jedoch Verhaltensbeschreibungen in das Modell integriert, geht die hier beschriebene Einbettung über das Chromosomenmodell hinaus und umfaßt auch eine ‚design language' im Sinne von [Mortensen, Andreasen 96], [Mortensen 97].

3.1 ‚design units' und ‚domains'

Das Alphabet dieser Sprache bilden ‚design units'. In den beiden letztgenannten Veröffentlichungen wird dieser Begriff noch nicht näher definiert, daher soll hier auf die Definition aus [Jensen, Hansen 98] verwiesen werden.

Die Eigenschaften der Abgeschlossenheit und der definierten Schnittstellen machen das Konstruktionsobjekt zur geeigneten informationstechnischen Umsetzung der ‚design unit'. Dabei entspricht eine Klasse von Konstruktionsobjekten einer ‚master unit'.

‚design units' enthalten Attribute, das sind Merkmale (‚characteristics'), die vom Konstrukteur definiert werden, und Eigenschaften (‚properties'), die aus den Merkmalen abgeleitetet werden und zur Beurteilung der Konstruktion dienen. Beide sind in den Daten der Konstruktionsobjekte enthalten.

‚design units' modellieren Produkte auf drei Ebenen (‚domains'): Funktion bzw. Transformation, Organ und Einzelteil. Daher werden drei Basisklassen „Funktion", „Organ" und „Teil" definiert. Jedes Konstruktionsobjekt, das eine ‚design unit' repräsentiert, wird von einer dieser Klassen abgeleitet.

3.1.1 ‚Function/Transformation unit'

Eine Funktion kann rein verbal beschrieben werden. Das entsprechende Konstruktionsobjekt enthält dann nur einen Text. Interessanter ist es jedoch, auch die Funktionen formal zu beschreiben, um auch auf dieser Ebene Methoden zu Durchführung von Berechnungen einsetzen zu können. Eine solche Formalisierung könnte sich an [Weber 86] anlehnen.

In beiden Fällen besitzt ein Konstruktionsobjekt vom Typ „Funktion" Schnittstellen für Subjekt(e) und Objekt(e) bzw. Ein- und Ausgabe sowie Relationen, die es in der Funktionsstruktur einordnen und mit den zugeordneten Organen verbinden. Dabei können auch Merkmale der Funktion auf solche der Organe abgebildet werden, wodurch eine automatische Synchronisierung erreicht wird.

3.1.2 ‚Organ unit'

Ein Organ enthält Schnittstellen zu anderen Organen, wobei die Gliederung in Rezeptoren und Effektoren, wie sie in [Mortensen, Andreasen 96] gefordert ist, nicht zwingend notwendig erscheint. Es ist denkbar, für unterschiedliche kausale Richtungen im selben Organ jeweils entsprechende Methoden zu integrieren.

Die Verbindung zwischen Funktionen und Organen wird durch die Relation „bedingt durch" realisiert, die Verbindung zwischen Organen und Teilen durch die Relation „wird realisiert durch".

3.1.3 ‚Part unit'

In [Mortensen, Andreasen 96] werden in bezug auf [Hubka 88] für die Repräsentation eines Teils vier Klassen von Merkmalen vorgeschlagen: Form, Material, Dimension und Oberflächenbeschaffenheit. Ligo selbst verwaltet keine explizite Geometrie, die über eine bloße

Visualisierung hinausgeht. Es begnügt sich mit einer parametrischen Darstellung, aus der mit Hilfe der entsprechenden Methoden die explizite Geometrie abgeleitet werden kann (dazu wird eine Eingabedatei für ein CAD-System erzeugt und dieses automatisch über einen Kommandozeilenaufruf gestartet).

Ob eine Beschränkung auf diese vier Eigenschaftsklassen sinnvoll ist, sollte in der Praxis überprüft werden, da mitunter auch organisatorische Eigenschaften wie Bestellnummern berücksichtigt werden müssen.

3.2 Detaillierungsgrade von Konstruktionsobjekten

Nach der ‚Theory of Domains' [Andreasen 80] werden Modelle entlang zweier Achsen „abstrakt-konkret" und „einfach-komplex" näher bestimmt. Konkreter wird ein Modell dabei durch Hinzufügen von Eigenschaften, komplexer durch Aufgliedern in Teile.

Da Ligo Unterobjekte und Daten formal gleich behandelt, fallen in der vorgeschlagenen Einbettung diese beiden Achsen zusammen. Detailliert wird grundsätzlich durch den Mechanismus des Ableitens, d.h. das komplexere Modell enthält durch den Vererbungsmechanismus das einfachere Modell vollständig - wobei aber Schnittstellen und Methoden durch speziellere, die den gleichen Namen tragen, überschrieben werden können.

3.3 Die Rolle von Plug-Ins

3.3.1 Realisierung von ‚Views'

Ein ‚View' enthält einige Eigenschaften des Modells selbst und einige abgeleitete Eigenschaften, die nur für diese Sicht interessant sind. Diese zusätzlichen Eigenschaften (und die Methoden zu ihrer Bestimmung) werden mit einem Plug-In hinzugefügt und damit Bestandteil des Produktmodells. Die einzelnen Plug-Ins lassen sich ein- und ausblenden, so daß der Benutzer nur diejenigen Eigenschaften sieht, an denen er gerade interessiert ist.

3.3.2 Nachträgliche Abstraktion von Konstruktionsobjekten

Da Plug-Ins für Basisklassen geschrieben werden können, ist in eine Bibliothek von generischen ‚Views' möglich, die bei Bedarf zu Klassen hinzugefügt werden können, die von diesen Basisklassen abgeleitet sind. Diese generischen ‚Views' können nötigenfalls ihrerseits zu spezielleren ‚Views' abgeleitet werden, die an spezielle Konstruktionsobjektklassen angepaßt sind. Dies hat den Vorteil, daß solche Plug-Ins für weitere Plug-Ins Funktionalitäten zur Verfügung stellen können, die konstruktionsobjektunabhängig sind.

3.3.3 kontextabhängige Eigenschaften (‚relational properties')

Manche Eigenschaften eines technischen Systems sind abhängig von Systemen außerhalb des betrachteten Modells. So sind Herstellkosten z.B. gleichzeitig vom Produkt und dem Produktionssystem abhängig. Wenn Modelle des Produktionsprozesses und des Produktionssystems in Form von Konstruktionsobjekten vorhanden sind, können speziell auf die Modelle von Produkt und Produktionssystem zugeschnittene Plug-Ins dem Modell des Konstruktionsprozesses über festgelegte Schnittstellen die Informationen aus beiden Modellen zur Verfügung stellen, die zur Berechnung der Herstellkosten benötigt werden. Genau genommen sind auch diese Plug-Ins ‚Views'.

3.3.4 Plug-Ins und Features

Nach [Femex 96] ist ein Feature nicht als abgeschlossenes Modell zu verstehen, sondern benötigt genau wie ein Plug-In einen Bezug zu einem existierenden Modell. In [Andreasen, Mortensen 96a] werden verschiedene Anwendungen von Features in einem Chromosomenmodell vorgestellt, die sich größtenteils durch Plug-Ins realisieren lassen. Ein Plug-In kann daher als ein Feature betrachtet werden, das seine Semantik (in Form von Methoden) selbst enthält.

4 Literatur

[Andreasen 80] M. M. Andreasen, „Machine Design Methods Based on a Systematic Approach – Contribution to a Design Theory, Diss., Department of Machine Design, Lund Institute of Technology, Sweden, 1980.

[Andreasen, Mortensen 96a] M. M. Andreasen, N. H. Mortensen, „The Nature of Design Features“, 7. Symposium „Fertigungsgerechtes Konstruieren“, Schnaittach, 1996

[Femex 96] C. Weber, „What is a Feature and What is ist Use?“, Results of FEMEX Working Group I, Proceedings of the 29th International Symposium on Automotive Technology and Automation (ISATA 96), pp. 287-296, Florenz, 1996

[Ferreirinha 90] P. Ferreirinha et al., „TEKLA, A Language for Developing Knowledge Based Design Systems“, Proceedings of ICED 90, Dubrovnik, Heurista., 1990

[Hubka, Eder 88] V. Hubka, W. E. Eder, „Theory of Technical Systems“, Springer-Verlag, Berlin, 1988

[Jensen, Hansen 98] T. Jensen, C.T. Hansen, „Capturing and Reuse of Design Knowledge during the Conceptual Design Process: Illustrated with a Snap-Fit Joint“, Proceedings of CACD'98, Lancaster, May, 1998

[Mortensen, Andreasen 96] N.H. Mortensen, M.M. Andreasen, „Designing in an Interplay with a Product Model – Explained by Design Unis“, TCME '96, Budapest

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

[Weber 86] Weber, C.: „Ein Beitrag zur integralen Betrachtungsweise von methodischem Konstruieren und Maschinenelementen.“, MeKoMe-Workshop 1986. WDK 14, pp.78ff, Heurista, 1986.

[Werner, Weber 98] H. Werner, C. Weber: „Ligo - an Object-Oriented Modelling Tool for Integrated Product Development“, 2nd International Workshop „Integrated Product Development“, Magdeburg, 1998.

Dipl.-Ing. Horst Werner, Prof. Dr.-Ing. Christian Weber
Lehrstuhl für Konstruktionstechnik/CAD
Universität des Saarlandes
Im Stadtwald, , Geb. 8.2, D 66041 Saarbrücken
Telefon: 0681-302 3387, Fax: 0681- 302 4858
e-mail: werner@cad.uni-sb.de