Projektbibliothek auf der Basis von Strukturplänen

Siegfried Wendt*

Stichworte: *Projektbibliothek, Softwareproduktion, diskrete Struktur, Dokument, CAD.*

Zusammenfassung: *Die Projektbibliothek als Werkzeug für die Aufgaben der Dokumentation und Projektüberwachung bei der rechnergestützten Softwareproduktion kann durch geeignete Abstraktion unter einem gemeinsamen Aspekt mit Werkzeugen aus dem Bereich des CAD für die Ingenieurwissenschaften gesehen werden. Unter konsequenter Beachtung dieser Gemeinsamkeiten läßt sich ein Werkzeug spezifizieren, welches so flexibel ist, daß es sich an eine breite Palette von Anwendungsfällen anpassen läßt und nicht von vornherein auf bestimmte Modellierungsschemata oder Projektmanagementstrukturen festgelegt ist. Die Strukturpläne als zweidimensionale symbolische Darstellungen diskreter Strukturen und die Korrespondenzen zwischen solchen Plänen bilden dabei den Schwerpunkt der Betrachtung.*

A project library based on charts of discrete structures

Key-words: *project library, software production, discrete structure, document, CAD.*

Abstract: *The project library as a tool for documentation and project control for software engineers can be viewed by appropriate abstraction in such a way that it fits into a common scheme with tools for CAD in other engineering disciplines. Strictly based on these common aspects, a tool is specified which is flexible enough to be adapted to a wide variety of application areas and which is not inherently restricted to certain techniques for system modelling or project management. The argumentation centers on the formal charts and the relationships between them, where a formal chart is a two-dimensional symbolic representation of a discrete structure.*

* Prof. Dr.-Ing. Siegfried Wendt, Fachbereich Elektrotechnik der Universität, Erwin Schrödinger Straße, D-6750 Kaiserslautern

1 Einleitung

Die Notwendigkeit der Rechnerunterstützung beim Entwurf technischer Komponenten oder Systeme ist heute unbestritten. Je nach der Art der zu entwerfenden Gebilde kommen unterschiedliche Werkzeuge der Rechnerunterstützung zum Einsatz. Die Abkürzung CAD/CAM erfaßt die Werkzeuge für Entwurf und Fertigung von Bauteilen oder Baugruppen aus den Bereichen Bauwesen, Maschinenbau und Elektrotechnik. Die Abkürzungen SPU (Software-Produktionsumgebung) oder SEE (Software-Engineering-Environment) erfassen die Werkzeuge der Rechnerunterstützung für die Softwareerstellung [5]. Die beiden Bereiche CAD/CAM und SPU bzw. SEE sind isoliert entstanden und gewachsen. Während CAD/CAM ohne den Einsatz von Grafik undenkbar ist, ist der Bereich SPU bzw. SEE bisher weitestgehend alphanumerisch orientiert. Erst in letzter Zeit ist das Bemühen stärker geworden, auch den Bereich SPU bzw. SEE mehr an der Grafik zu orientieren. Diesem Bemühen gilt auch die vorliegende Arbeit.

Wenn man die Werkzeuge des Bereiches SPU bzw. SEE nach der Grafik ausrichtet, dann wird die scharfe Abgrenzung gegenüber dem Bereich CAD/CAM durchbrochen, denn dann werden bestimmte Analogien sichtbar. Man erkennt dann, daß die Systemklasse der sogenannten Netzsysteme eine Klammer zwischen beiden Bereichen bildet; es gehören ihr nämlich nicht nur Elemente aus dem Bereich CAD/CAM an – beispielsweise elektrische Netze, logische Schaltungen oder Gasversorgungsnetze, sondern auch alle Softwaresysteme, denn diese können durch entsprechende Modellierung stets als Netzsysteme betrachtet werden. Die zugehörigen grafischen Darstellungen sind Strukturpläne, d.h. zweidimensionale Anordnungen von Symbolen zur Veranschaulichung diskreter mathematischer Strukturen. Da das Sichtbarmachen unterschiedlicher Strukturaspekte komplexer Softwaresysteme eine grundlegende Voraussetzung für die ingenieurmäßige Beherrschung solcher Softwaresysteme ist, wird die Bereitstellung entsprechender Werkzeuge immer dringlicher. Die vorliegende Arbeit stellt das Konzept eines solchen Werkzeugs vor. Dieses Werkzeug kann vorgestellt werden, ohne daß eine Methode zur Sichtbarmachung von Softwarestrukturen eingeführt wird, weil nämlich dieses Werkzeug nur durch die abstrakte Welt

0013-5704/84/11 0452–07 $ 03.00/0

der diskreten Strukturen bestimmt wird und nicht durch den Sonderfall der Softwarestrukturen. Deshalb sind die dargestellten Beispiele nur als Anschauungshilfen im Zusammenhang mit abstrakten Begriffsdefinitionen zu sehen und nicht als Ausprägungen einer Methode, obwohl sie natürlich die Richtung andeuten, in der nach Ansicht des Autors solche Methoden liegen [12].

2 Aufgaben einer Projektbibliothek

Unter einer Projektbibliothek versteht man ein Werkzeug zur Unterstützung der Softwareproduktion, welches die Funktionen eines Produktverwaltungssystems und eines Informationssystems für das Projektmanagenemt vereinigt [2].

Produktverwaltung bedeutet in diesem Zusammenhang die retrieval-effiziente und konsistenzüberwachte Speicherung sämtlicher Informationsobjekte, die als Gesamtheit das Softwareprodukt darstellen. Die Menge der Informationsobjekte reicht dabei von der Anforderungsspezifikation über die Quellcode-Module bis zum Benutzerhandbuch. Die Forderung nach Retrieval-Effizienz bedeutet, daß das System in der Lage sein soll, das jeweilige suchzielorientierte Wissen des Rechercheurs möglichst vollständig in den Suchanfragen zu akzeptieren und zur Führung eines optimalen Retrieval-Dialogs auszuwerten. Im primitivsten Fall akzeptiert das System nur die direkte Identifikation von Informationsobjekten mittels eindeutiger Bezeichner, im Falle höchster Effizienz dagegen akzeptiert das System in der Suchfrage beliebige Aussagen über Inhalte von Informationsobjekten und über inhaltliche Beziehungen zwischen Informationsobjekten. Die Forderung nach Konsistenzüberwachung bedeutet, daß das System in der Lage sein soll, bei der Annahme neuer Informationsobjekte im Dialog mit dem Anbieter der Objekte dafür zu sorgen, daß stets nur aussagenverträgliche Mengen von Informationsobjekten gespeichert werden. Die Forderung nach Aussagenverträglichkeit kann vom technischen System allerdings nur für Aussagen mit formaler Semantik befriedigt werden; für die Überwachung der Verträglichkeit anderer Aussagen bleibt der Systembenutzer zuständig.

Der Funktionsanteil einer Projektbibliothek, der diese zu einem Informationssystem für das Projektmanagement macht, ist gekennzeichnet durch die Begriffe Planung, Überwachung und Steuerung. Planung bedeutet Vorgabe von zu erbringenden Leistungen in Form von Arbeitsergebnissen und jeweils zugestandenem Aufwand. Überwachung bedeutet den dauernden Vergleich der Planungswerte mit den im Projektverlauf sich ergebenden Istwerten, und Steuerung bedeutet die Änderung der Planungswerte oder der Rahmenbedingungen bei Abweichungen der Istwerte von den bisherigen Planungswerten.

Eine Projektbibliothek kann die Planung nur unterstützen, wenn Analogien zu früheren Projekten erkannt und ausgenutzt werden. Dies erscheint nur mit Methoden der Künstlichen Intelligenz – Stichwort Expertensysteme – realisierbar. Dagegen reichen die heute üblichen Methoden vollständig aus, die Überwachungsfunktion in einer Projektbibliothek zu realisieren. Die Steuerung schließlich kann nur dadurch unterstützt werden, daß aus den Überwachungsergebnissen Hinweise auf mögliche Schwachstellen der Planung oder der Rahmenbedingungen abgeleitet werden; auch in diesem Fall kann der Rückgriff auf Informationen aus früheren Projekten nützlich sein.

3 Pläne für diskrete Strukturen

Der folgenden Betrachtung von Mengen und Relationen liegt die elementare Begriffsbildung der diskreten Mathematik zugrunde, wie sie auch von CHEN [1] als Basis eines allgemeinen Datenbegriffs vorgeschlagen wurde. Jede der hier betrachteten diskreten Strukturen ist definiert als eine endliche Anzahl m disjunkter endlicher Mengen M_1, M_2, ... M_m zusammen mit einer endlichen Anzahl r von Relationen R_1, R_2, ... R_r, die zwischen den Mengen bestehen. Jede Relation R_j ist als Teilmenge eines kartesischen Produkts der Dimension d_j aufzufassen, dessen Faktoren $F_{j,k}$ jeweils die Vereinigung einer Auswahl der gegebenen Mengen M_i sind:

$R_j \subseteq F_{j,1} \times F_{j,2} \times F_{j,3} \times \ldots \times F_{j,d_j}$

mit $\phi \subset F_{j,k} \in$ Potenzmenge von $\{M_1, M_2, \ldots M_m\}$

Beispiele:

$R_1 \subseteq M_1 \times M_1$; $d_1 = 2$

$R_2 \subseteq (M_2 \cup M_4) \times M_2 \times M_5 \times (M_3 \cup M_4)$; $d_2 = 4$

$R_3 \subseteq M_5 \times (M_5 \cup M_3 \cup M_4) \times M_5$; $d_3 = 3$

Jeder Menge M_i und jeder Relation R_j ist ein eigener Attributvektor $V_{b.n}$ zugeordnet, wobei b die binäre Klassenkennzeichnung M oder R ist und n der Index i von M_i bzw. j von R_j. Die Dimension v von $V_{b,n}$ hängt von b und n ab.

$V_{b.n} = (A_{b.n.1}, A_{b.n.2}, \ldots, A_{b.n.v})$

Unterschiedlich indizierte Attributvektoren dürfen gleich sein.

Jedem Attribut $A_{b.n.p}$ ist ein Attributtyp T_q zugeordnet, der gleich einer Menge möglicher Attributwerte $W_{q.s}$ ist:

$T_q = \{W_{q.1}, W_{q.2}, \ldots, W_{q.u}\}$

Die Mächtigkeit u der Menge T_q hängt von q ab. Unterschiedlich indizierte Attribute $A_{b.n.p}$ dürfen denselben Typ T_q haben.

Jedem Element $E_{b.n.e}$ von M_i bzw. R_j ist eine Wertebelegung $B_{b.n.e}$ des zugehörigen Attributvektors $V_{b.n}$ zugeordnet. Eine solche Wertebelegung besteht darin, daß jeder Komponente $A_{b.n.p}$ des Attributvektors $V_{b.n}$ ein Attributwert $W_{q.s}$ des zugehörigen Attributtypes T_q zugeordnet wird.

Damit sind alle Elemente einer diskreten Struktur im hier gemeinten Sinne formal eingeführt. Sie müssen nun durch ein Beispiel veranschaulicht werden. Dazu wird Bild 1 betrachtet. In diesem Beispiel gibt es vier Mengen M_i

$M_1 = \{I1, I2, I3, I4\}$
$M_2 = \{a, b, c, d\}$
$M_3 = \{x, y\}$
$M_4 = \{\alpha, \beta, \gamma, \delta, \epsilon\}$

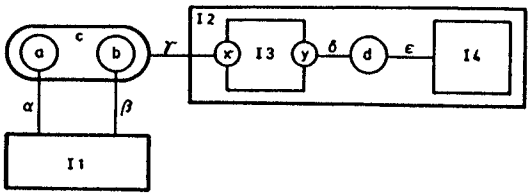

Bild 1 Beispiel einer grafisch dargestellten diskreten Struktur

und drei jeweils zweistelligen Relationen:

$R_1 \subset (M_1 \cup M_2) \times (M_1 \cup M_2 \cup M_3 \cup M_4)$
$R_2 \subset M_3 \times M_1$
$R_3 \subset (M_1 \cup M_2 \cup M_3) \times M_4$.

Die Relationen drücken folgende Beziehungen zwischen den Elementen $E_{M.i.e}$ der Mengen M_i aus:

R_1: Ein Element aus M_1 oder M_2 umschließt ein Element aus einer der Mengen M_1 bis M_4.

R_2: Ein Element aus M_3 liegt auf dem Rand eines Elements aus M_1.

R_3: Ein Randpunkt eines Elements aus M_1 oder M_2 oder M_3 ist ein Endpunkt eines Elements aus M_4.

Damit ergeben sich die Relationen als folgende Mengen geordneter Paare:

R_1 = {(I2, I3), (I2, I4), (I2, d), (I2, x), (I2, y), (I2, δ), (I2, ϵ), (c, a), (c, b)}
R_2 = {(x, I3), (y, I3)}
R_3 = {(I1, α), (I1, β), (I4, ϵ), (a, α), (b, β), (c, γ), (d, δ), (d, ϵ), (x, γ), (y, δ)}

Bild 2 stellt eine Verbindung zwischen den Elementen in Bild 1 und der Welt der Software her: Die Elemente von M_1, also die Rechtecke, sind abwickelbare Programmstücke; die Elemente von M_2, also die großen Kreise oder Ovale, sind Speicherzellen für Programmvariable; die Elemente von M_3, also die auf Rechteckrändern liegenden kleinen Kreise, sind formale Variable von Prozeduren; die Elemente von M_4, also die Verbindungslinien, sind Zugangspfade.

```
Modul C
    Exportierte Zugriffsrechte:   zu a, b, c;
    Deklarierte Variable:         c bestehend aus a, b;

Modul I1
    Importierte Zugriffsrechte:   α zu a, β zu b;
    Rest von Modul I1;

Modul I2
    Importiertes Zugriffsrecht:   γ zu c;
    Deklarierte Variable:         d
    Deklarierte Prozedur:         I3(x, y);
                                  Rest der Prozedur;

    Rest von Modul I2
        bestehend aus I4
        und möglicherweise mehrfachem CALL I3(c, d);
```

Bild 2 Programm in Pseudocode als mögliche Quelle von Bild 1

Ein Strukturplan, wie er in Bild 1 vorliegt, zeigt eine diskrete Struktur ohne Attribute, d.h. ein solcher Strukturplan erlaubt nur die Aufzählung der einzelnen Elemente der Mengen M_i und R_j, aber abgesehen von der durch die grafischen Symbole ausgedrückten Klassifikation macht der Plan keine Aussagen über Eigenschaften der aufgezählten Elemente. Diese müssen also getrennt durch Angabe der Attributvektoren und ihrer Wertebelegungen vermittelt werden. Als Beispiel sei der Attributvektor $V_{M.1}$ der Menge M_1, also der Rechtecke in Bild 1, betrachtet für den Fall, daß diese Rechtecke Funktionseinheiten sind, die durch abwickelbare Programme bestimmt sind. Der Phantasie sind keine Grenzen gesetzt bei der Festlegung eines zugehörigen Attributvektors, der die verschiedenen Einzelinformationen zusammenfaßt, die man zu einem Rechteck in Bild 1 assoziieren will. Sicher muß der Programmcode selbst dazugehören, aber welche darüberhinausgehende Information man noch aufnehmen will, kann nur durch Zweckmäßigkeitsüberlegungen bei der Spezifikation einer Projektbibliothek entschieden werden. Willkürlich wird folgender Beispielsvektor vorgestellt:

$V_{M.1} = (A_{M.1.1}, A_{M.1.2}, A_{M.1.3}, A_{M.1.4}, A_{M.1.5})$

mit $A_{M.1.1}$ = Name der Funktionseinheit
$A_{M.1.2}$ = Name des zuständigen Entwicklers
$A_{M.1.3}$ = Geplanter Fertigstellungstermin
$A_{M.1.4}$ = Bearbeitungszustand
$A_{M.1.5}$ = Programmcode

Als Beispiel für einen Attributtyp T_q wird der zu $A_{M.1.4}$ gehörende Typ vorgestellt, der hier willkürlich als T_2 bezeichnet wird:

$T_2 = \{W_{2.1}, W_{2.2}, W_{2.3}, W_{2.4}\}$

mit $W_{2.1}$ = geplant
$W_{2.2}$ = geplant und spezifiziert
$W_{2.3}$ = geplant, spezifiziert und codiert
$W_{2.4}$ = geplant, spezifiziert, codiert und getestet

So wie man einen Entwurfsgegenstand im Bereich des CAD/CAM meist durch mehrere Pläne unterschiedlichen Typs erfaßt – z.B. Grundrisse und Elektroinstallationspläne eines Gebäudes –, die aufeinander bezogen sind, so ist es auch im Bereich SPU bzw. SEE selbstverständlich, daß es Pläne unterschiedlichen Typs gibt, die auf verschiedene Weisen aufeinander bezogen sein können. Obwohl es sich auch um Relationen im bereits definierten Sinne handelt, soll für die Beziehungen zwischen Plänen und zwischen Elementen auf verschiedenen Plänen im folgenden die Bezeichnung „Korrespondenzen" verwendet werden, damit deutlicher abgegrenzt wird, ob von den strukturellen Beziehungen innerhalb eines Plans oder von anderen die Rede ist. Um diese Aussagen zu veranschaulichen, wird mit Bild 3 ein weiterer Strukturplan vorgestellt, wobei Korrespondenzen zwischen diesem Plan und dem in Bild 1 bestehen. Bild 3 zeigt ein Petrinetz, welches in sequentielle Teilnetze zerlegt ist, die den Programmen der vier Funktionseinheiten in Bild 1 entsprechen. Diese Korrespondenz K_1 kann formal geschrieben werden als

$K_1 \subset$ {Teilnetze aus Bild 3} x {Rechtecke aus Bild 1}

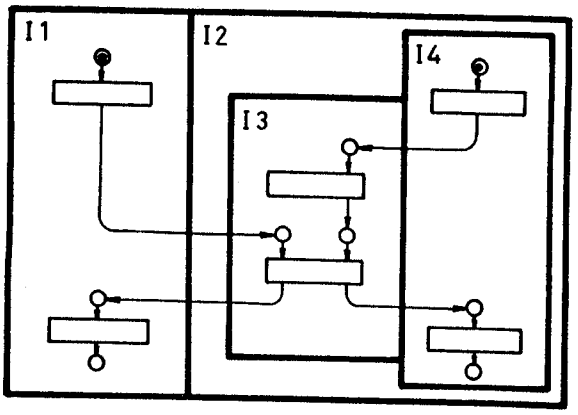

Bild 3 Petrinetz in Korrespondenz zu Bild 1

Zwischen Korrespondenzen und Relationen bestehen häufig formale Abhängigkeiten, die es erlauben, aus gegebenen Korrespondenzen und Relationen weitere Korrespondenzen oder Relationen abzuleiten. Dieser Fall liegt auch im betrachteten Beispiel vor: Die Relation, welche die Teilnetze in Bild 3 definiert und formal als

$R_4 \subset$ {Alle Elemente aus Bild 3} x {Teilnetze aus Bild 3}

zu schreiben ist, legt zusammen mit K_1 eine weitere Korrespondenz K_2 fest:

$K_2 \subset$ {Alle Elemente aus Bild 3} x { Rechtecke aus Bild 1}

Dadurch wird folgender Sachverhalt zum Ausdruck gebracht: Jedes Element in Bild 3, das über R_4 bestimmten Teilnetzen zugeordnet ist, ist über K_2 auch denjenigen Rechtecken in Bild 1 zugeordnet, welche über K_1 seinen Teilnetzen zugeordnet sind.

Grundsätzlich können auch Korrespondenzen in gleicher Weise wie Relationen durch Attributvektoren attributiert werden, aber es gibt praktisch meistens keine Gründe dafür.

Korrespondenzen lassen sich in drei unterschiedliche Klassen einteilen, nämlich in modellierungsmittelbedingte, modellierungsinhaltsbedingte und systemänderungsbedingte Korrespondenzen.

Modellierungsmittelbedingte Korrespondenzen ergeben sich, weil man i.a. mehrere Strukturpläne teilweise unterschiedlichen Typs braucht für die vollständige Darstellung eines als diskrete Struktur modellierten Systems. So liefern beispielsweise die Strukturpläne in den Bildern 1 und 3 unterschiedliche Beiträge zur Darstellung eines einzigen diskreten Systemmodells; Bild 1 sagt etwa aus über die Aufbaustruktur, Bild 3 dagegen über die Ablaufstruktur. Auch Detaillierung führt zu modellierungsmittelbedingten Korrespondenzen, welche als Enthaltenseinskorrespondenzen zu interpretieren sind. Zwei Strukturpläne sind immer dann über modellierungsmittelbedingte Korrespondenzen verbunden, wenn sie Beiträge zur Darstellung ein und desselben Systemmodells liefern.

Dagegen sind zwei Strukturpläne über modellierungsinhaltsbedingte Korrespondenzen verbunden, wenn sie Beiträge zur Darstellung unterschiedlicher Modellierungen ein und desselben Systems liefern. In der Sprache der Schaltungstechnik heißt dies, daß unterschiedliche Ersatzschaltbilder ein und desselben realen Systems zueinander in Beziehung gebracht werden. Als Beispiele dafür, daß solche Korrespondenzen auch in der Softwaredokumentation auftreten können, sei ein Programmsystem aus der Prozeßtechnik betrachtet: Einerseits wird man dies als ein System modellieren, worin mehrere Aktionen nebenläufig auftreten können; andererseits muß man die auf einem Monoprozessor notwendige Sequentialisierung dieser Aktionen auch modellieren, so daß man am Ende zwei unterschiedliche Modellierungen ein und desselben Systems hat.

Systemänderungsbedingte Korrespondenzen verbinden Strukturpläne gleichen Typs, die zwei ähnliche Systeme unter dem gleichen Aspekt darstellen. Solche Korrespondenzen treten i.a. bei jedem Projekt auf, denn sie erfassen die Entwurfsänderungen bzw. -korrekturen, die sich im Laufe eines Projekts aufgrund neuer Einsichten ergeben. Jede Struktur- oder Attributänderung, falls sie eine Änderung des erfaßten Systems bedeutet, bringt die Gegenüberstellung eines älteren und eines neueren Plans mit sich, zwischen denen systemänderungsbedingte Korrespondenzen bestehen. Auch bei der Betrachung einer Systemfamilie, deren Mitglieder sich als unterschiedliche Konfigurationen aus einem Repertoire von Bausteinen ergeben, findet man zwangsläufig Korrespondenzen dieser Kategorie.

4 Projektmanagement auf der Basis von Strukturplänen

Durch die Wahl des Beispiels zum Attributvektor wurde im vorigen Abschnitt bereits angedeutet, daß man bei der Festlegung des Schemas einer Projektbibliothek nicht nur die Produktverwaltung, sondern auch die Belange des Projektmanagements berücksichtigen muß. Es ist offensichtlich, daß man sehr viel managementrelevante Informationen als Attribute an produktbezogene Strukturelemente anbinden kann – wie im Beispiel den Namen des zuständigen Entwicklers, den geplanten Fertigstellungszeitpunkt oder den Bearbeitungszustand. Darüberhinaus gibt es allerdings noch andere managementrelevante Informationen, die sich nicht als Attribute produktbezogener Strukturelemente unterbringen lassen. Man kann diese jedoch stets als Elemente in eigens dafür definierten Strukturplänen erfassen, wobei diese Pläne dann untereinander und zu produktbezogenen Strukturplänen Korrespondenzen haben können. Indem man die Information für das Projektmanagement auf diese Weise in die Welt der diskreten Strukturen aufnimmt, schafft man sich den großen Vorteil, bei der Gestaltung des Werkzeugkerns für die Projektbibliothek nicht unterscheiden zu müssen zwischen Produktverwaltung und Projektmanagement, sondern sich vielmehr ganz auf die abstrakte Welt der diskreten Strukturen konzentrieren zu können.

5 Funktionen und Aufbau des Werkzeugs

Wenn hier von der Projektbibliothek als einem Werkzeug gesprochen wird, dann ist damit ein Satz mehrerer zusammengehöriger Werkzeuge gemeint. Bild 4.1 zeigt, daß jedes Mitglied einer Projektgruppe über einen solchen Werkzeugsatz verfügt und damit auf lokalen Daten, globalen Daten und Off-line-Daten operieren kann. Um die Nebenläufigkeit unverträglicher Datenzugriffe zu verhindern, darf den Werkzeugen kein direkter Datenzugriff möglich sein; deshalb greifen sie auf die Daten nur durch Vermittlung einer zentralen Instanz zu. Die Unterteilung der Daten im On-line-Archiv in lokale und globale Daten bringt zum Ausdruck, daß jeder Benutzer einerseits einen eigenen Datenbestand hat, auf dem er uneigeschränkt operieren kann, und daß er andererseits eingeschränkten und überwachten Zugang zu einem globalen Datenbestand und möglicherweise auch zu lokalen Datenbeständen anderer Benutzer hat. Die globalen Daten stellen den konsistenzüberwachten Inhalt der Projektbibliothek dar.

Der wesentliche Systemaspekt, der durch das Aufbaustrukturmodell in Bild 4.1 sichtbar wird, ist die Mehrbenutzereigenschaft, aus der die Notwendigkeit einer zentralen Zugriffskoordinationsinstanz folgt. Dieses Modell eignet sich aber nicht dazu, die in der Funktion der einzelnen Werkzeuge begründeten Zugriffsbeziehungen zwischen Werkzeugen und Daten sichtbar zu machen. Dies gelingt jedoch mit dem Aufbaustrukturmodell in Bild 4.2, welches in modellierungsinhaltsbedingter Korrespondenz zu Bild 4.1 steht. In Bild 4.2 haben die verschiedenen Werkzeuge eines Benutzers direkten Zugriff auf die jeweils betroffenen Daten, d.h. die Frage, wie die Zugriffskoordinaten realisiert werden soll, wird hier als irrelevant betrachtet. Der wesentliche Systemaspekt, der durch dieses Aufbaustrukturmodell in Bild 4.2 sichtbar gemacht wird, ist die Klassifikation der verschiedenen Werkzeuge eines einzelnen Benutzers nach ihrer Funktion und ihren dadurch bedingten Datenzugängen. Der Benutzer kann jeweils nur mit einem einzigen Werkzeug arbeiten; die Existenz der Umschaltemöglichkeit ist gezeigt, die Frage ihrer Realisierung aber wurde als irrelevant betrachtet.

Die umfassenden Datenbestände sind partitioniert in Komponenten, die teilweise einer Erklärung bedürfen. Strukturpläne werden durch ihr Vorkommen in Dokumenten erfaßt, wobei zwischen Originalvorkommen und Kopiervorkommen unterschieden wird. Bei Kopiervorkommen bleibt das Editieren auf Maßstabsveränderungen und Positionsveränderungen des ganzen Planes beschränkt. Kopiervorkommen gibt es häufig dadurch, daß Pläne aus Entwicklungsdokumenten in Benutzer- oder Wartungsanleitungen übernommen werden. Es wird zwischen umgebrochenen und umbrechbaren Dokumenten unterschieden: Das Layout eines umgebrochenen Dokumentes kann, im Gegensatz zum umbrechbaren Dokument, bei der Ausgabe nicht beeinflußt werden. Entwicklungsdokumente, die nur Strukturpläne enthalten, sind typischerweise umgebrochen, während i.a. eine Benutzeranleitung umbrechbar gespeichert ist. Das Umbrechen ist Aufgabe der Ein/Ausgabe- und Wandelwerkzeuge. Da auf einem Dokument mehrere Pläne vorkommen dürfen, gibt es nicht nur Korrespondenzen zwischen Dokumenten, sondern auch innerhalb eines Dokuments. Während aber die Relationen als Strukturinformation in Form der Pläne Bestandteil der Dokumenteninformation sind, werden die Korrespondenzen zwischen Plänen auf einem einzigen Dokument nicht als diesem Dokument entnehmbare Information angesehen, sondern als Information, die außerhalb des Dokuments festgehalten werden muß.

Der Auftrennung der globalen Daten in relationale Daten und Attribut-Dateien liegt folgende Vorstellung zugrunde: In einer relationalen Datenbank lassen sich alle Mengen M_i, Relationen R_j und Korrespondenzen K_k retrieval-

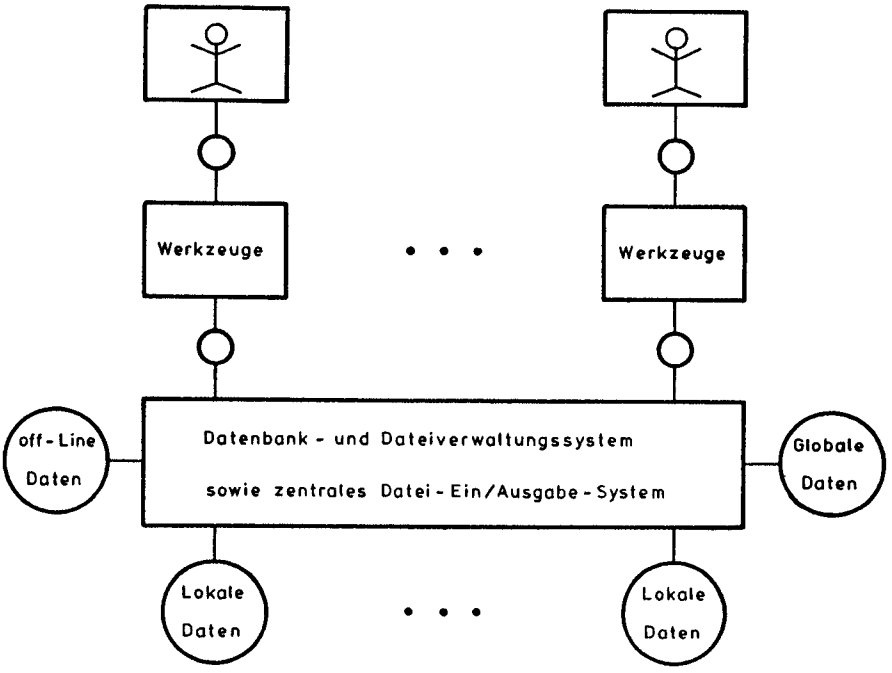

Bild 4.1
Aufbaustruktur der Projektbibliothek in erster Modellierung

effizient unterbringen. Die Wertebelegungen $B_{b.n.e}$ der Attributvektoren dagegen lassen sich in die relationale Datenbank nur dann vollständig einbringen, wenn alle Attributwerte $W_{q.s}$ bestimmte vorgegebene Formatbedingungen erfüllen. Dies ist im Falle der Projektbibliothek nicht gegeben, d.h. hier gibt es immer bestimmte Attributtypen T_q, deren Werte die Formatbedingungen nicht einhalten. Zur Speicherung von Attributwerten solcher Typen überführt man deshalb T_q in T'_q, indem man jeden Wert $W_{q.s}$ durch Hinzufügen eines eindeutigen Bezeichners $I_{q.s}$ in $W'_{q.s}$ überführt:

$$W'_{q.s} = (I_{q.s}, W_{q.s})$$

Auch wenn die Menge T_q unendlich sein mag, bleibt die Teilmenge der abzuspeichernden Werte aus T_q immer endlich, so daß man die Elemente dieser Teilmenge immer unter Einhaltung der Formatbedingungen der relationalen Datenbank bezeichnen kann. Deshalb lassen sich alle Werte $I_{q.s}$ in der relationalen Datenbank unterbringen als eindeutige Verweise zum Auffinden der als formatfreie Dateien gespeicherten Attributwerte $W_{q.s}$.

Jedes Dokument in den globalen Daten ist einerseits ein logisches Element der relationalen Daten und andererseits, in Form der editierten Datei, ein formatfreies Attribut dieses Elements. Da in dieser Datei sowohl die Strukturinformation als auch die Layoutinformation der vorkommenden Pläne enthalten sind, die Strukturinformation aber auch zu den relationalen Daten gehört, ist also die Strukturinformation doppelt gespeichert. Dies ist sinnvoll, da man die Strukturinformation sowohl zum Retrieval und zur Konsistenzhaltung beim Update als auch zum Editieren der Dokumente benötigt.

Auf die Werkzeuge kann im Rahmen dieser Arbeit leider nur sehr kurz eingegangen werden. Als oberstes Gebot bei der Spezifikation der Werkzeuge gilt, daß sich die im Ablaufcode festgelegten Funktionen möglichst nur auf die abstrakte Welt der diskreten Strukturen beziehen sollen und daß das Werkzeugverhalten für die verschiedenen konkreten Fälle – beispielsweise elektrische Netze, logische Schaltungen, Systemmodelle mit Petrinetzen, SARA [3], IORL [9] oder andere Modellwelten [4], [6], [11] – möglichst nur durch Vorgabe entsprechender Parameterdaten in Form von Modellweltbeschreibungen festgelegt wird. Dieses Gebot läßt sich nur beim Dokumenteneditor, der die gemischte Bearbeitung von logisch strukturierten Grafik- und Textkomponenten erlaubt, und beim Retrieval-Werkzeug vollständig einhalten. Für die anderen Werkzeuge dagegen bringen die verschiedenen Anwendungsfälle derart unterschiedliche Anforderungen bezüglich der Benutzerschnittstelle, daß hier eine Anpas-

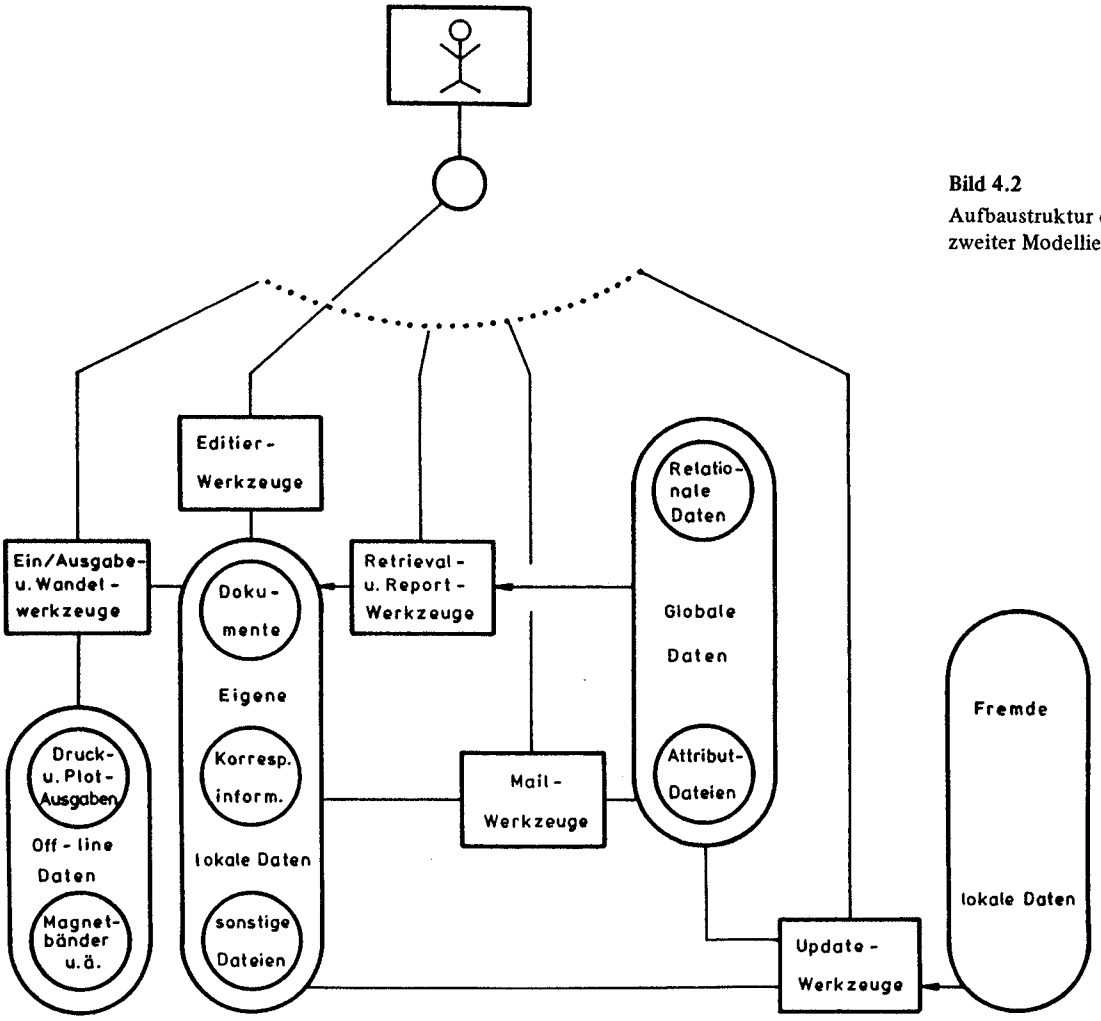

Bild 4.2
Aufbaustruktur der Projektbibliothek in zweiter Modellierung

sung ohne teilweisen Modulaustausch nicht möglich ist. Die Reportwerkzeuge dienen einerseits dazu, Auszüge aus den relationalen Daten zu gewinnen und diese in benutzerfreundliche Form zu bringen, andererseits werden sie in Verbindung mit Wandelwerkzeugen eingesetzt, um Information aus den globalen Daten für Werkzeuge außerhalb der Projektbibliothek – z.B. Simulatoren oder Analysatoren – aufzubereiten. Die Update-Werkzeuge dienen dazu, die globalen Daten konsistenzüberwacht zu verändern, wobei entweder Daten hinzugefügt oder herausgenommen werden. Hinzuzufügende Daten kommen entweder direkt über das Terminal vom Benutzer oder aus den eigenen oder fremden lokalen Daten. Das Herausnehmen, d.h. das Löschen von globalen Daten ist ein besonders kritischer Vorgang, der nur unter Einhaltung strenger Regeln und nur von Benutzern mit besonderen Hoheitsrechten durchgeführt werden kann. Dabei können lokale Dateien entstehen, in denen die herausgenommenen Daten aufgefangen werden, damit man sie an Online- oder Offline-Datenhaltungssysteme außerhalb der Projektbibliothek weitergeben kann.

Die Mail-Werkzeuge dienen dazu, die projektbezogene Kommunikation innerhalb der Projektgruppe zu unterstützen und sie bezüglich bestimmter, für das Projektmanagement relevanter Botschaftstypen zu erzwingen. Auch die Mail-Werkzeuge können auf der Basis diskreter attributierter Strukturen konzipiert und realisiert werden; die Briefkästen und ihre Inhalte sind Elemente der globalen Daten.

Ein Prototyp einer solchen Projektbibliothek wird zur Zeit am Institut für Digitalsysteme der Universität Kaiserslautern implementiert. Der Host-Rechner ist eine VAX 11-750 unter VMS; die Programmiersprache ist C. Als relationale Datenbank wird das DEC-System DATATRIEVE verwendet, welches durch ein spezielles Frontend eine Schnittstelle für die Anwendungsprogramme erhielt, deren Semantik durch die Welt der diskreten Strukturen bestimmt ist [7]. Das Farbrasterterminal für die Edition der Dokumente ist eine Eigenentwicklung, die besonders nach den Belangen des Editierens ausgerichtet wurde und die deshalb das Erstellen der Editor-Software stark vereinfacht [13].

6 Schlußbetrachtung

Während bei der Betrachtung von Projektbibliotheken üblicherweise das zugrundeliegende „life cycle"-Modell und die Datenbanktechnik in den Vordergrund gestellt werden, liegt in der vorliegenden Arbeit die Betonung auf den Gemeinsamkeiten mit anderen CAD-Werkzeugen und auf der Strukturierung des Werkzeugs unter dem Gesichtspunkt der Flexibilität. Denn das Gebiet SPU bzw. SEE kann noch keineswegs als abgeklärt gelten, und das letzte Wort über die optimale Gestaltung von Projektbibliotheken ist noch nicht gesprochen [8]. Deshalb ist es sinnvoll zu versuchen, eine Projektbibliothek so zu gestalten, daß sie sich auch zum Experimentieren mit unterschiedlichen Modellierungstechniken und Managementstrukturen eignet. Ein bekannter Ansatz, der in diese Richtung zielt, ist PSL/PSA [10]. Um weiterzukommen muß man sich jedoch von der dortigen rein alphanumerischen Orientierung lösen und auch von der dort vorgegebenen Klassifikation der Mengen und Relationen abstrahieren. Die vorliegende Arbeit kann als Ergebnis eines solchen Schrittes gewertet werden.

Der DFG sei an dieser Stelle gedankt für die teilweise Förderung unserer Arbeiten zu diesem Thema.

Literatur

[1] *Chen, P. P.:* The Entity-Relationship Model-Toward a Unified View of Data. In: ACM Transactions on Database System 1/1976, S. 9–36

[2] *Denert, E., Hesse, W.:* Projektmodell und Projektbibliothek: Grundlagen zuverlässiger Software-Entwicklung und Dokumentation. In: Informatik-Spektrum 3/1980, S. 215–228

[3] *Estrin, G.:* A Methodology for design of Digital Systems – Supported by SARA at the age of One. In: Proc. of National Computer Conference 1978, AFIPS Press

[4] *Freeman, P.:* A Perspective in Requirements Analysis and Specification. In: Infotech State of the Art Report on Structured Software Development, 1978

[5] *Hausen, H.-L.:* Software-Produktionsumgebungen.In: Informatik Spektrum 1/1983, S. 39–40

[6] *Luft, A. L.:* Zur Bedeutung von Modellen und Modellierungsschritten in der Software-Technik. In: Angewandte Informatik 5/1984, S. 189–196

[7] *Rösner, W.:* Das Datengittermodell und seine Konsequenzen für den Entwurf informationsverarbeitender Systeme mit großen Datenbasen. Dissertation, Universität Kaiserslautern, 1983

[8] *Schnupp, P.:* Wie wirklich ist die Software-Technologie? In: GI – 13. Jahrestagung 1983, Informatik-Fachberichte Bd 73, Springer-Verlag, Berlin, 1983, S. 161–173

[9] IORL Version 2, Language Reference Manual. Teledyne Brown Engineering, Huntsville, Alabama, 1982

[10] *Teichroew, D., Hershey, E. A.:* PSL/PSA: A computer aided technique for structured documentation and analysis of information processing systems. In: IEEE Transactions on Software Engineering, 1/1977

[11] *Trauboth, H.:* An Engineering View of System Specification and Design Tools. In: Infotech State of the Art Report on Structured Software Development, 1979

[12] *Wendt, S.:* Der Kommunikationsansatz in der Softwaretechnik. In: SIEMENS data report 4/1982, S. 4–7

[13] *Wendt, S.:* Einführung in das Grafiksystem GRILL. Bericht 82 B-12, 2. Aufl., FB Elektrotechnik, Universität Kaiserslautern, 1984.