1. CAD IM BAUWESEN

Zeichnungen werden seit der Antike bei der Planung, Ausführung und Nutzung von Bauwerken eingesetzt. Bis vor wenigen Jahren besaßen diese Zeichnungen einen hauptsächlich dokumentierenden Charakter. In maßstabsgerechten Abbildungen und in schematischen Darstellungen werden die Geometrie, die visuelle Wirkung, die Konstruktion, das Verhalten und die Verwendung von Bauwerken so mit graphischen Elementen dargestellt, daß verschiedene Betrachter die Zeichnung mit möglichst gleichem Ergebnis semantisch interpretieren. Die Semantik der Zeichnungen ergibt sich aus der Formähnlichkeit bei Abbildungen, der Reduktion auf wesentliche Merkmale (beispielsweise Kanten), der Definition graphischer Symbole und der Anordnung graphischer Elemente relativ zueinander auf dem Zeichnungsträger.

Entwicklungsstadien : Heute sind Zeichnungen im Computer mit Modellen hinterlegt. Die Modelle der frühen Jahre des Computer Aided Design (CAD) sind Mengen graphischer Elemente, die in digitaler Verschlüsselung gespeichert werden. Sie lassen sich einfacher ändern als Zeichnungen auf Papier. Die semantische Interpretation der graphischen Elemente als Bauteile und Eigenschaften von Bauwerken bleibt Aufgabe des Betrachters.

Die weitere Entwicklung des CAD ist durch zunehmende Strukturierung der graphischen Elementmengen gekennzeichnet. Ein frühes Beispiel ist die Gliederung der Elemente in Zeichnungsebenen, die semantisch interpretiert und mit eigenen Darstellungsmerkmalen versehen werden [1]. Von besonderer Bedeutung ist der Übergang zu objekt-orientierten Modellen [2]. An die Stelle der aus graphischen Elementen zusammengesetzten Zeichnungen treten aus allgemeinen Objekten zusammengesetzte Bauwerke oder Bauprozesse als primäre Informationsbasis [3]. Visualisierungsmodelle werden nach Bedarf aus den Bauwerksmodellen abgeleitet. Objektklassen haben semantische Bedeutung und kennen selbst ihre Darstellungsformen in verschiedenen graphischen Umgebungen. Mit diesem Übergang vom dokumentenorientierten Ansatz zum modellorientierten Ansatz wird die Zeichnung direkt mit anderen Aspekten des Ingenieurprozesses verbunden, so mit der Berechnung des Verhaltens von Bauwerken, mit ihrer Konstruktion und Ausführung und mit der Verwaltung ihrer Nutzung.

Informationsbasis : Mit der Objektstruktur von Bauinformationen wachsen die Ansprüche an die Organisation der digitalen Informationsbasis [4,5,6]. Erste Ansätze zur Übergabe von Objekten an standardisierten Schnittstellen (Ports) sind wegen der inkompatiblen Struktur der vorhandenen Bau-Software gescheitert. Die Standardisierung des semantischen Objektmodells für das Bauwesen ist wichtig [7], steht jedoch erst am Anfang einer allgemeinen Einführung im Bauwesen. Die Erfahrung zeigt, daß Softwaremodule in der Regel getrennt entwickelt und vereint genutzt werden. Ziel der aktuellen Forschung und Entwicklung ist es daher, die speziellen Informationsbasen unabhängig entwickelter Module zum Zeitpunkt der Nutzung interaktiv zu interpretieren und unter aktiver Mitwirkung der fachlich kompetenten Nutzer zusammenzuführen [8]. Dieser aktive Integrationsansatz ergänzt den rechnerintern gesteuerten Datenfluß im semantischen Objektmodell.

Organisation : Die Regeln für die Nutzung einer digitalen Informationsbasis müssen vorab festgelegt und für alle Beteiligten verbindlich sein [4]. Die einfachste Organisationsform ist die Ergänzung des Versands traditioneller technischer Dokumente durch den Versand digitaler Dateien. Hierfür sind die internen Datenformate oder Schnittstellenformate, die Kennzeichnung neuer, geänderter und gelöschter Informationen in Versionen [5] sowie die Urheber- und Zugriffsrechte festzulegen. Ein weiterer Ansatz ist die Organisation eines zentralen Datenpools für sämtliche in einem Projekt erstellten Informationen. Festzulegen sind Regeln für die Archivierung und Verteilung von digitalen Datenbeständen und Dokumenten sowie für deren Beschreibung (Eigentümer, Freigabebestand, digitales

Format,...). Mit zunehmender Leistungsfähigkeit der Computer-Netzwerke kann die Zentralisierung auf einen Informationskatalog beschränkt werden, während die Information selbst bei Bedarf von dezentralen Netzservern abgerufen wird (Holeschuld der Nutzer).

Focus : In der aktuellen Entwicklung des CAD ist ein zentrales Problem erkennbar, das auf alle anderen Aspekte der Entwicklung ausstrahlt. Dies ist die Beschreibung und Handhabung der Beziehungen zwischen Objekten und der Änderungen in strukturierten Objektmengen. Eine effiziente Lösung dieser Aufgabe ist Voraussetzung für die Entwicklung von Bausoftware für vernetzte, kooperative Arbeitsumgebungen und für die Einführung neuer Kommunikationsprozesse im Bauwesen. Dieser Aufsatz soll die Grundzüge der Aufgabenstellung mit der Relationenalgebra als theoretischer Grundlage [9] zeigen; die Entwicklung leistungsfähiger Methoden und Algorithmen für die beschriebene Aufgabe ist Gegenstand aktueller Forschung.

Die Grundzüge dieser Aufgabenstellungen betreffen vor allem die strukturierte und formalisierte Definition der Objektbeziehungen für CAD und die theoretisch fundierte Verfolgung von Änderungen an Objekten. Beide Aspekte haben vor allem auch für CAD im Bauwesen eine große Bedeutung, denn bisherige Vorgehensweisen beruhen in aller Regel auf verfügbaren pragmatischen Mechanismen. Die Verfasser sind der Überzeugung, daß vor allem der Update-Prozeß getrennt bearbeiteter Informationsmengen im Bauprozeß von fundamentaler Bedeutung für die Nutzbarkeit zukünftiger CAD-Systeme sein wird und daß dieser daher wesentlicher Bestandteil der Grundkonzepte sein muß.

2. ABHÄNGIGKEIT ZWISCHEN OBJEKTEN

Applikation : Die Menge der Klassen, Attribute, Methoden und Objekte, die zur Lösung einer bestimmten Aufgabe mit dem Computer eingesetzt werden, heißt eine Applikation. Die Beziehungen zwischen den Objekten einer Applikation werden mit den folgenden Begriffen beschrieben.

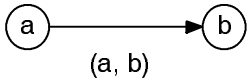
Bindung : Ein Objekt b heißt an ein Objekt a gebunden, wenn mindestens eine der folgenden Bedingungen erfüllt ist :

- Methoden des Objektes a ändern Attribute des Objektes b oder der Klasse von b.
- Methoden des Objektes b verwenden Attribute des Objektes a oder der Klasse von a.

Bindungsrelation : Die Menge der Objekte einer Applikation sei M. Ist ein Objekt $b \in M$ an ein Objekt $a \in M$ gebunden, so ist das geordnete Paar (a,b) ein Element der Bindungsrelation $B \subseteq M \times M$:

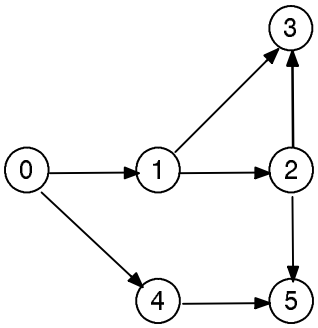
$$B := \{ (a, b) \in M \times M \mid a \text{ bindet } b \}$$

Eine Bindungsrelation wird geometrisch als Graph und algebraisch als boolesche Matrix dargestellt. Im Graphen wird ein Objekt als ein Knoten und eine Bindung (a,b) als eine von Knoten a nach Knoten b gerichtete Kante dargestellt.



Die Darstellung der Bindungsmatrix **B** für eine Menge M mit n Objekten erfordert die Anordnung der Objekte durch eine Abbildung $f : N \rightarrow M$ mit $f(i) = a_i$ und Index $i \in \{0, 1,...,n-1\}$. Die Abbildung f ist nicht eindeutig. Ihre Wahl beeinflußt den Aufwand in den Algorithmen der Objektverwaltung. Dem Objekt i werden die Zeile i und die Spalte i von **B** zugeordnet. Der Koeffizient b_{im} von B ist true (integer 1), wenn (a_i, a_m) ein Element der Bindungsrelation B ist. Sonst besitzt b_{im} den Wert false (integer 0).

Beispiel 1 : Bindungsrelation eines Graphen ohne Zyklen



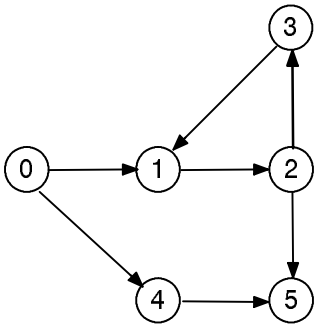
D =

0	1			1		0
	0	1	1			1
		0	1		1	2
			0			3
				0	1	4
					0	5
0	1	2	3	4	5	

Objektkette : Eine Objektmenge {a,...,z} heißt eine Objektkette und wird mit [a,b,...,y,z] bezeichnet, wenn kein Objekt mehr als einmal in der Kette enthalten und jedes Objekt durch seinen Vorgänger gebunden ist.

Objektzyklus : Eine Objektmenge {a,...,z} heißt ein Zyklus und wird mit $<$a,b,...,y,z$>$ bezeichnet, wenn kein Objekt mehr als einmal in der Kette enthalten, z der Vorgänger von a und jedes Objekt durch seinen Vorgänger gebunden ist.

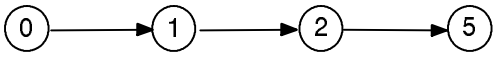
Beispiel 2 : Bindungsrelation eines Graphen mit Zyklus $<$1, 2, 3$>$



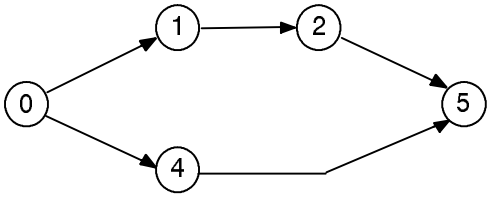
D =

0	1			1		0
	0	1				1
		0	1		1	2
	1		0			3
				0	1	4
					0	5
0	1	2	3	4	5	

Abhängigkeit : Ein Objekt b $\in$ M heißt abhängig von einem Objekt a $\in$ M, wenn es mindestens eine Objektkette [a,x_1,...,x_m,b] gibt, die mit a anfängt und mit b endet. Diese Abhängigkeit wird mit $|a,b|$ bezeichnet. Gebundene Elemente sind abhängig, also (a,b) = $|a,b|$. Im Graphen wird eine Abhängigkeit durch einen Weg von Knoten a nach Knoten b dargestellt, beispielsweise von Knoten 0 nach Knoten 5 in Beispiel 2.



Zwischen abhängigen Objekten kann es mehr als eine Kette geben, beispielsweise zwischen den Objekten 0 und 5 in Beispiel 2. Diese Ketten können verschieden lang sein :



Kontextrelation : Die Menge der Objekte einer Relation sei M. Ist ein Objekt b ∈ M von einem Objekt a ∈ M abhängig, so ist das geordnete Paar (a,b) ein Objekt der Kontextrelation H ⊆ M x M :

$$H := \{ (a,b) \in M \times M \mid b \text{ ist von a abhängig } \}$$

Die Kontextrelation H ist die transitive Hülle der Bindungsrelation B. Sie spezifiziert die Abhängigkeit zwischen Objekten. Ihre Bestimmung ist eine bekannte Aufgabe der Graphentheorie. Die Kontextrelation wird analog zur Bindungsrelation als Graph oder als boolesche Kontextmatrix **H** mit den Koeffizienten h_{im} dargestellt. Da die Potenzmatrix B^s alle Wege der Länge s im Graphen enthält, ist H die Vereinigung von höchstens n - 1 Potenzmatrizen.

$$H = B \cup B^2 \cup \ldots \cup B^{n-1}$$

Kontextalgorithmus : Das folgende Java-Fragment zeigt einen Algorithmus zur Bestimmung der Kontextmatrix **H** für eine gegebene Bindungsmatrix **B**. Die Hülle von B wird zeilenweise in einer Schleife über i aufgebaut. Dazu wird zunächst die Zeile i von **B** nach **H** kopiert. Dann wird in Zeile i von **H** ab Spalte 0 ein Element mit dem Wert true gesucht. Enthält Spalte k den Wert true, so ist das Objekt a_k von Objekt a_i abhängig. Damit sind auch alle Objekte, die von Objekt a_k abhängen, von Objekt a_i abhängig. Die Zeile i von **H** wird daher mit der Zeile k von **B** vereinigt. Die Spalte k von **H** wird markiert : Bei nachfolgenden Durchsuchungen der Zeile i wird die Spalte k übersprungen, da die Zeile k von **B** bereits in der Hülle erfaßt ist. Die Zeile i von **H** wird solange ab Spalte 0 durchsucht, bis kein unmarkiertes Element mit Wert true mehr vorhanden ist. Damit sind alle Knoten bekannt, die von Knoten i abhängen. Der Algorithmus ist auch für Graphen mit Zyklen geeignet.

```
class Context
{ boolean[][] link      ;                  // Bindungsmatrix
  boolean[][] hull      ;                  // Kontextmatrix
  int         dimension ;                  // Anzahl Knoten

  void computeHull()
  { boolean hit, mark[]                                        ;
    int     m,row,col                                          ;

//....Berechne eine Zeile der Hülle.........................
    mark = new boolean[dimension]                              ;
    hull = new boolean[dimension][dimension]                   ;
    for (row = 0; row < dimension; row++)
    { System.arraycopy(link[row],0,hull[row],0,dimension)      ;
      for (m=0; m < dimension; m++) mark[m] = false            ;
      mark[row] = true                                         ;
//....Suche ein unmarkiertes Element mit Wert true............
      do
      { hit= false                                             ;
        for (col = 0; col < dimension; col++)
        { if (hit = (hull[row][col] == true) && !mark[col])
          break                                                ;
        }
//....Vereinigung der Zeilen hull[row] und link[col].........
        if(hit)
        { mark[col] = true                                     ;
          for (m = 0; m < dimension; m++)
          { hull[row][m] |= link[col][m]                       ;
      } } }
```

```
      while(hit) ;
} } }
```

Komplexität : Die Berechnung der Kontextmatrix **H** für eine Menge von n Objekten als Vereinigung von Potenzen der Bindungsmatrix **B** besitzt die Komplexität $0(n^4)$. Die Komplexität des Kontextalgorithmus ist hingegegen $0(n^3)$. Der Algorithmus kann durch Verwendung von Bitoperationen weiter beschleunigt werden.

Beispiel 3 : Kontextmatrizen $\mathbf{H}_1$ und $\mathbf{H}_2$ für die Beispiele 1 und 2

0	1	1	1	1	1	0
0	0	1	1	0	1	1
0	0	0	1	0	1	2
0	0	0	0	0	0	3
0	0	0	0	0	1	4
0	0	0	0	0	0	5
0	1	2	3	4	5	$\mathbf{H}_1$

0	1	1	1	1	1	0
0	1	1	1	0	1	1
0	1	1	1	0	1	2
0	1	1	1	0	1	3
0	0	0	0	0	1	4
0	0	0	0	0	0	5
0	1	2	3	4	5	$\mathbf{H}_2$

3. ÄNDERUNGEN VON OBJEKTEN

Update : Die Berechnung der abhängigen Attribute von Objekten einer Applikation aus gegebenen Attributwerten der Objekte selbst sowie anderer Objekte der Applikation heißt ein Update (eine Aktualisierung) der Applikation. In einem Update werden nicht unbedingt alle abhängigen Objekte der Applikation aktualisiert.

Zielmenge eines Update : Eine Applikation kann auf mehr als eine Weise aktualisiert werden. Beispielsweise kann bei einer Änderung der Geometrie eines Bauwerkes entweder nur die graphische Darstellung oder nur die Massenermittlung oder die graphische Darstellung und die Massenermittlung aktualisiert werden. Folglich besitzt eine Applikation im allgemeinen verschiedene Updates, die zu verschiedenen Zeiten ausgeführt werden können. Die Objekte, deren Attribute in einem bestimmten Update neu zu berechnen sind, bilden die Zielmenge Z dieses Update.

$$Z := \{ a \in M \mid \text{a wird im Update neu berechnet} \}$$

Änderungsmenge : Ein Objekt kann durch den Anwender, durch seine eigenen Methoden oder durch Methoden anderer Objekte oder Klassen geändert werden. Ein Objekt heißt geändert, wenn eines seiner Attribute oder ein Attribut seiner Klasse geändert wurde. Die geänderten Objekte werden in der Änderungsmenge A der Applikation vermerkt.

$$A := \{ a \in M \mid \text{a wurde geändert} \}$$

Reduktion der Änderungsmenge : Nach einer Aktualisierung der Applikation muß die Änderungsmenge A ebenfalls aktualisiert werden. Es können jedoch nicht alle Objekte in der Änderungsmenge A gelöscht werden, für die abhängige Objekte in dem Update aktualisiert wurden : Es kann nämlich andere Updates geben, die noch nicht ausgeführt wurden und deren Zielmenge weitere Objekte enthält, die ebenfalls von den geänderten Objekten abhängen.

Das Löschen von Objekten in der Änderungsmenge nach Ausführung eines Update heißt eine Reduktion der Änderungsmenge. Zur Steuerung der Reduktion wird ein Reduktionsverzeichnis angelegt. Für jedes Objekt, das in mindestens einer Update-Zielmenge enthalten ist, sind im Verzeichnis alle Updates vermerkt, in deren Zielmenge das Objekt enthalten ist. Datentechnisch kann das Reduktionsverzeichnis beispielsweise als HashMap (key,value) implementiert werden. Für ein Objekt mit Identifikator id, das in Updates mit den Identifikatoren u_0, u_1,...u_j, u_k auftritt, enthält das Verzeichnis folgende Eintragungen :

$(key_0, value_0)$:= (id, u_0)

$(key_1, value_1)$:= $(id+u_0, u_1)$

. .

. .

. .

$(key_k, value_k)$:= $(id+u_j, u_k)$

Zur Verwaltung des Verzeichnisses werden mehrere Methoden benötigt :

void	**put** (objectId, updateId)	:	Eintragung
void	**remove** (objectId, updateId)	:	Austragung
boolean	**contains** (objectId)	:	Abfrage
Iterator	**iterator** (objectId)	:	Traverse der Updates mit **next** ( )

4. REIHENFOLGE DER OBJEKTE IM UPDATE

Anordnung : Ist ein Objekt $b \in M$ von einem Objekt $a \in M$ abhängig, das geändert wurde, so muß das Objekt a vor dem Objekt b aktualisiert werden. Da es zwischen a und b mehrere Objektketten geben kann, ist die Reihenfolge der Objekte im Update nicht offensichtlich. Objektzyklen verursachen besondere Schwierigkeiten. Daher werden zunächst nur zyklenfreie Graphen betrachtet. Die Zeitachse des Update wird mit fortlaufend ab 0 indizierten Stationen in Intervalle zerlegt. Jeder Station ist eine Menge von Objekten zugeordnet, die in beliebiger Reihenfolge aktualisiert werden dürfen. Der Index der Station heißt das Alter der Objekte dieser Menge. Junge Objekte werden vor alten Objekten aktualisiert. Eine Abbildung $g : M \rightarrow S$ einer Objektmenge M auf eine Stationsmenge S heißt eine Anordnung von M. Im folgenden wird ein Algorithmus zur Bestimmung einer zulässigen Anordnung g für eine Menge M mit gegebener Bindungsrelation B entwickelt.

Domäne : Die Zielmenge eines Update enthält diejenigen Objekte der Applikation, deren Attribute in diesem Update zu aktualisieren sind. Ein Objekt a_m der Zielmenge kann von mehreren Objekten a_i,...,a_k abhängig sein. Diese Objekte werden in Spalte m der Kontextmatrix der Applikation gelesen. Ist a_m beispielsweise von a_j abhängig, so muß Objekt a_j beim Aufstellen der Anordnung des Update berücksichtigt werden, wenn a_j ein Objekt der Änderungsmenge ist. Die Teilmenge der Objekte der Änderungsmenge A, die von mindestens einem Objekt der Zielmenge Z des Update abhängig sind, heißt die Domäne des Update und wird mit D bezeichnet. Die Domäne enthält den Durchschnitt $A \cap Z$ als Teilmenge, da die geänderten Objekte der Zielmenge von sich selbst abhängen.

$$D := \{ a \in A \mid \bigvee_{z \in Z} (z \text{ hängt von a ab}) \}$$

Nur die Objekte der Domäne werden im Update aktualisiert. Die nicht in der Domäne enthaltenen Objekte der Zielmenge Z des Update bleiben also unverändert. Da die Objekte der Kette von einem Objekt der Änderungsmenge A zu einem Objekt der Zielmenge Z des

Update nicht notwendig in A oder in Z enthalten sind, müssen sie explizit in die Domäne aufgenommen werden. Gibt es zwischen diesen beiden Objekten mehrere Ketten, so sind alle zu berücksichtigen.

Kontext einer Domäne : Die Bindungsrelation einer Domäne ist eine Teilmenge der Bindungsrelation B der Applikation und wird mit $\hat{B}$ bezeichnet. Die Bindungsmatrix $\hat{\mathbf{B}}$ des Update enthält diejenigen Zeilen und Spalten der Bindungsmatrix **B** der Applikation, die Objekten der Domäne D zugeordnet sind. Die Kontextrelation der Domäne, die mit $\hat{H}$ bezeichnet wird, ist eine entsprechende Teilmenge der Kontextrelation H der Applikation.

$$\hat{B} := \{ (a,b) \in D \times D \mid a \text{ bindet } b \}$$

$$\hat{H} := \{ (a,b) \in D \times D \mid b \text{ ist in D von a abhängig} \}$$

Chronomatrix : Die Bestimmung einer Anordnung für die Objekte einer Domäne D mit gegebener Bindung $\hat{B}$ ist eine Aufgabe der Wegalgebra. Sie entspricht der Bestimmung des längsten Weges zwischen zwei Knoten eines Wegenetzes mit Kanten gleicher Länge 1. Gibt man den Elementen der Bindungsmatrix $\hat{\mathbf{B}}$ die integer-Werte 0 (false) und 1 (true), so ist der Koeffizient b^k_{im} der Potenzmatrix $\hat{\mathbf{B}}^k$ gleich 1, wenn es eine Kette der Länge k von Objekt a_i nach Objekt a_m gibt. Das Alter der Knoten a_i und a_m muß um mindestens k Intervalle differieren. Da der Bindungsgraph voraussetzungsgemäß keine Zyklen enthält, ist die maximale Kettenlänge kleiner als die Anzahl der Objekte der Domäne. Diese Länge heißt Stabilitätsindex der Kontextmatrix $\hat{\mathbf{B}}$ und wird mit s bezeichnet.

Der maximal erforderliche Altersunterschied von Objekt a_i nach Objekt a_m wird als Koeffizient der Chronomatrix **T** der Domäne gespeichert. Zur Berechnung der Chronomatrix werden die Potenzen $\hat{\mathbf{B}}^k$ der Bindungsmatrix mit dem Integer k multipliziert und kombiniert. Der Operator + symbolisiert das Maximum von kb^k_{im} über k. Zwischen zwei Objekten der Domäne kann es nämlich mehrere Ketten geben. Der Koeffizient b^k_{im} kann daher für mehrere Potenzen k gleich 1 sein. Maßgebend für den erforderlichen Altersunterschied zwischen den Objekten a_i und a_m ist die maximale Potenz k, für die b^k_{im} gleich 1 ist :

$$\mathbf{T} := \hat{\mathbf{B}}^1 + 2\hat{\mathbf{B}}^2 + \ldots + s\hat{\mathbf{B}}^s$$

$$t_{im} := \max_k \left(k\, b^k_{im} \right)$$

Der maximale Altersunterschied v_m zwischen dem Objekt a_m und allen anderen Objekten der Domäne ist das Maximum der Altersunterschiede t_{im} über die Objekte a_i, von denen a_m abhängt. Das Alter der Objekte, die von keinem anderen Objekt der Domäne abhängen, wird auf 0 gesetzt. Dann besitzt der allgemeine Knoten a_m das Alter v_m .

Altersalgorithmus : Das folgende Java-Fragment zeigt einen Algorithmus zur Bestimmung der Altersunterschiede t_{im} und des Alters v_m für eine gegebene Bindungsmatrix $\hat{\mathbf{B}}$ der Domäne. Als Anfangswerte der Koeffizienten t_{im} von **T** werden die Altersunterschiede der gebundenen Objektpaare eingesetzt, also $t_{ii} = 0$, $t_{ik} = 1$ für b_i == true und t_{ik} = NONE (keine Bindung) für b_{ik} == false.

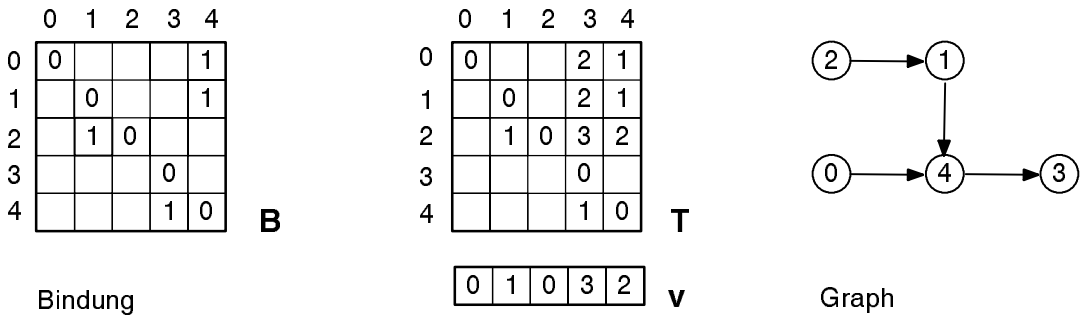
Die maximalen erforderlichen Altersunterschiede t_{im} werden in drei geschachtelten Schleifen über k,i,m bestimmt [10]. In der äußeren Schleife über k werden alle Ketten gebildet, die durch Kettung einer bei a_i beginnenden und bei a_k endenden Kette mit einer bei a_k beginnenden und bei a_m endenden Kette entstehen. Besitzt t_{ik} oder t_{km} den Wert NONE (keine Bindung), so ist eine Kettung nicht möglich und t_{im} bleibt unverändert. Sonst wird die Länge der gebildeten Kette als Summe der Längen der Teilketten bestimmt. Sind a_i und a_m bisher noch nicht verbunden (t_{im} == NONE) oder ist die Länge der gebildeten Kette größer als der bisher maximale erforderliche Altersunterschied t_{im}, so wird t_{im} auf den Wert s gesetzt.

```
class Sequence
{ static final int NONE = -1 ;              // keine Bindung
  boolean[][] link           ;              // Bindungsmatrix
  int[][]     chronos        ;              // Altersunterschied
  int[]       age            ;              // Alter der Objekte
  int         dimension      ;              // Anzahl Objekte

  void computeAge()
  { int m,row,col,test ;
    chronos = new int[dimension][dimension] ;
//....Setze Anfangswerte des erforderlichen Altersunterschiedes....
    for (row = 0; row < dimension; row++)
    { for (col = 0; col < dimension; col++)
      { if (row == col)          chronos[row][col] = 0    ;
        else if (link[row][col]) chronos[row][col] = 1    ;
        else                     chronos[row][col] = NONE ;
    } }
//....Berechne den maximalen erforderlichen Altersunterschied......
    for (m = 0; m < dimension; m++)
    { for (row = 0; row < dimension; row++)
      { if (chronos[row][m] == NONE) continue ;
        for (col = 0; col < dimension; col++)
        { if (chronos[m][col] == NONE) continue ;
          test = chronos[row][m] + chronos[m][col] ;
          if (test > chronos[row][col]) chronos[row][col] = test ;
    } } }
//....Berechne das Alter der Objekte..............................
    age = new int[dimension] ;
    for (col = 0; col < dimension; col++)
    { test = 0 ;
      for (row = 0; row < dimension; row++)
      { if (test < (m = chronos[row][col])) test = m ;
      }
      age[col] = test ;
} } }
```

Komplexität : Die Berechnung der Chronomatrix **T** für eine Menge von n Objekten als Kombination von Potenzen $\hat{k}\mathbf{B}^k$ besitzt die Komplexität $0(n^4)$. Die Komplexität des Altersalgorithmus ist $0(n^3)$. Bei geringer Belegungsdichte kann die Suche nach Elementen $t_{ik} \neq NONE$ und $t_{km} \neq NONE$ mit referenzierten Datenstrukturen beschleunigt werden.

Beispiel 4 : Bestimmung des Objektalters



Für den Graphen mit 2 unabhängigen und drei abhängigen Objekten sind die Bindungsmatrix **B**, die Chronomatrix **T** und der Altersvektor **v** gezeigt.

5. ERWEITERUNGEN

In den Abschnitten 2 bis 4 ist ein Konzept für die Verwaltung strukturierter Objektmengen beschrieben. Die Anwendung dieses Konzeptes erfordert robuste Algorithmen, die auch für große und verteilt gespeicherte Objektmengen geeignet sind. Mit der Entwicklung solcher Algorithmen ist Forschung auf den folgenden Gebieten verbunden :

Zyklen in Graphen : Ingenieuraufgaben werden häufig interaktiv bearbeitet. Ihre Kontextgraphen enthalten folglich Zyklen. In einem Zyklus ist zwar die Reihenfolge der Objekte im Update bestimmt; die Anzahl der Bearbeitungsschritte ist jedoch im Gegensatz zu zyklusfreien Graphen nicht begrenzt. Zur Verwaltung solcher Objektmengen sind zwei zusätzliche Elemente erforderlich :

- In den Zyklus werden Kontaktobjekte eingebaut. Die Methoden eines Kontaktobjektes unterbrechen den Zyklus sobald ein gegebenes Konvergenzkriterium erfüllt ist. Damit wird die Anzahl der Bearbeitungsschritte begrenzt.
- Im Objektgraphen der Applikation wird der Zyklus durch ein Makroobjekt ersetzt. Das Makroobjekt ist mit jedem Knoten gebunden, mit dem eines seiner Objekte gebunden ist. Zur Aktualisierung des Makros wird sein Iterationszyklus ausgeführt, bis ein Kontaktobjekt den Zyklus unterbricht. Makros können andere Makros enthalten. Nach der Reduktion aller Zyklen zu Makros ist die Bindungsrelation der Applikation zyklenfrei.

Die Bestimmung von Zyklen in Graphen und die Zerlegung von Graphen in Komponenten sind bekannte Aufgaben der Graphentheorie [9]. Zyklen werden im Kontextalgorithmus daran erkannt, daß ein bereits markiertes Element h_{im} nochmals gefunden wird. Für den Ersatz von Zyklen durch Makros können verschiedene Datenstrukturen benutzt werden. Ihre Güte ist systematisch zu erforschen.

Beispiel 5 : Ersatz eines Zyklus durch ein Makroobjekt

Graph mit Zyklus

Graph mit Makro

Verteilte Applikationen : Eine Applikation heißt verteilt, wenn der Zugriff auf mindestens ein Objekt der Applikation durch mindestens ein anderes Objekt der Applikation den offenen oder versteckten Einsatz eines Kommunikationsdienstes erfordert. Es wird vorausgesetzt, daß der Zugriff auf ein Objekt über einen Kommunikationsdienst wesentlich aufwendiger ist als der Zugriff auf ein Objekt im lokalen Arbeitsspeicher. Die Datenbasen mittlerer und großer Projekte und Organisationen sind in der Regel verteilt gespeichert.

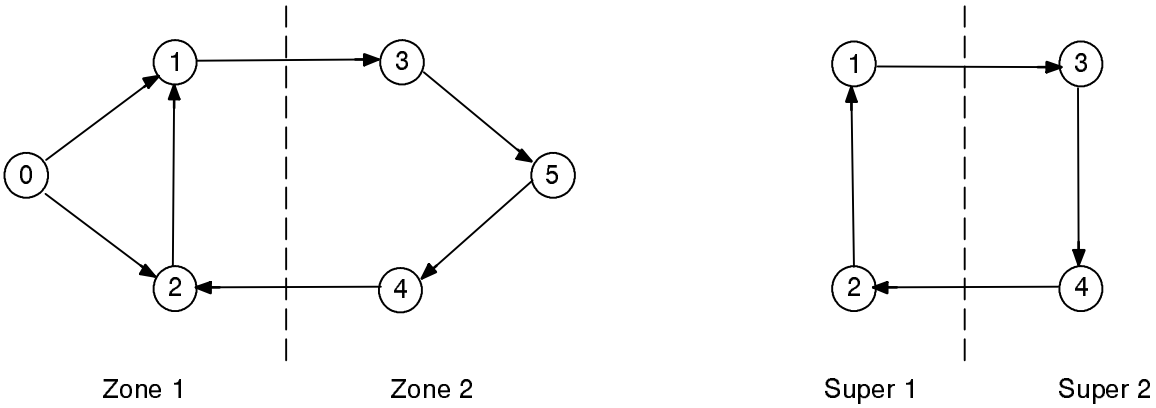
Der Kontext und das Alter einer lokalen Objektmenge können mit wesentlich geringerem Aufwand bestimmt werden als dieselben Eigenschaften einer globalen Objektmenge. Für große Objektmengen ist es daher zweckmäßig, zwischen lokalem und globalem Zusammenhang zu unterscheiden. Dabei ist zu beachten, daß zwei Objekte global abhängig sein können, obwohl sie lokal unabhängig sind.

Bei verteilt bearbeiteten Objektmengen ergibt sich zusätzlich zur Effizienz der Algorithmen noch ein weiterer wesentlicher Aspekt. Dieser betrifft die Komplexität der Verfolgung von Änderungen. Die Abhängigkeiten in der Bearbeitung von Objektmengen sind bei lokaler

Abspeicherung durchaus noch mit herkömmlichen Methoden überschau- und beherrschbar. Dies ist bei verteilt gespeicherten Objektmengen eindeutig nicht mehr der Fall. Hier sind formalisierte Verfahren und Vorgehensweisen unverzichtbar.

Lokale Objektmengen werden im globalen Graphen durch Superobjekte ersetzt. Dazu wird die lokale Objektmenge in innere Objekte und Randobjekte zerlegt. Ein Objekt heißt inneres Objekt, wenn es nur Abhängigkeiten zwischen diesem Objekt und anderen lokalen Objekten gibt. Die Bindungsmatrix des Superobjektes folgt aus der lokalen Kontextmatrix durch Streichung der inneren Objekte.

Beispiel 6 : Superobjekte einer verteilten Applikation



Im folgenden sind die Bindungsmatrix **B** der Zone 2, die Kontexmatrix **H** der Zone 2 und die Bindungsmatrix **S** des Superobjektes 2 gezeigt.

B =

	3	4	5
3	0	0	1
4	0	0	0
5	0	1	0

H =

	3	4	5
3	0	1	1
4	0	0	0
5	0	1	0

S =

	3	4
3	0	1
4	0	0

Dynamische Relationen : Im Vorangehenden wurde vorausgesetzt, daß die Bindungsmatrix einer Applikation konstant ist. Eine Applikation wird jedoch in der Regel schrittweise aufgebaut. Dabei ändert sich nicht nur die Objektmenge der Applikation. Häufig werden auch Bindungen ergänzt oder entfernt. Für die Bindungsmatrix, die Kontextmatrix, die Chronomatrix und den Altersvektor solcher Applikationen sind statische Datenstrukturen ungeeignet. Für die entsprechenden dynamischen Datenstrukturen werden Grundoperationen wie “Objekt eintragen” und “Objekt austragen” benötigt, die mit möglichst geringem Aufwand die abgeleiteten Relationen aktualisieren.

Mengenbildung : Die Semantik einer Applikation bewirkt häufig, daß vorab bekannt ist, ob die Änderung eines bestimmten Objektes ein Update erzwingt, in dem alle Objekte einer bestimmten Objektmenge zu berücksichtigen sind. Ändert man beispielsweise in einem Lastfall eine der Lasten, so gehen alle Lasten des Lastfalls in die Berechnung des neuen Verschiebungszustandes ein. Daher ist es zweckmäßig, die Objekte des Lastfalls nicht einzeln in die Bindungsrelation der Applikation aufzunehmen, sondern den Lastfall als ein Mengenobjekt zu behandeln. Dementsprechend wird in die Änderungsmenge der Applikation nicht die Einzellast aufgenommen, sondern der Lastfall. Diese Vorgehensweise setzt aber voraus, daß bekannt ist, in welchen Lastfällen eine bestimmte Last auftritt.

Die Definition zweckmäßiger Mengenobjekte kann den Umfang der Bindungsrelation einer Applikation wesentlich reduzieren. Beispiele solcher Mengenobjekte sind Knotenmengen, Stabmengen, Lastfälle und Lagerungsfälle in statischen Berechnungen. Für die Mengenobjekte können Versionen eingeführt werden, um die Änderung der Applikation mit der Zeit zu dokumentieren [11].

6. FOLGERUNGEN

Die systematische Behandlung der Beziehungen zwischen den Objekten einer CAD-Applikation sowie der Folgen von Änderungen der Objektattribute und der Bindungsrelationen durch Anwender sind ein wichtiger Aspekt der aktuellen Entwicklung, die vernetzt-kooperative Lösungen für Bauingenieuraufgaben anstrebt. Obwohl die fundamentalen Konzepte zur mathematischen Darstellung und Behandlung der Beziehungen und der Änderungen vorliegen, erfordern die großen Objektmengen praktischer Aufgabenstellungen und ihre Nutzung in verteilten Arbeitsumgebungen die Erforschung und Entwicklung neuer Methoden und Algorithmen. Einige der erforderlichen Arbeiten sind in diesem Aufsatz skizziert.

Literatur

[1] Abeln, O. (Hrsg.) : CAD-Referenzmodell
Teubner, Stuttgart, 1995.

[2] Haas, W. : CAD in der Bautechnik. Bauingenieur 63 (1988) 95 - 104,
Springer-Verlag, Berlin.

[3] Hartmann, D. (Hrsg.) : Objektorientierte Modellierung in Planung und Konstruktion. Forschungsbericht. Deutsche Forschungsgemeinschaft.
Wiley-Verlag, Weinheim (2000).

[4] Beucke, K. : Einfluß der Informationstechnik auf den Bauablauf.
Management / Baubetriebswirtschaft 12/99, Seite 40 - 43.

[5] Ilieva, D.; Pahl, P.J. : Eine Datenbahn für das Bauen. Abschlußbericht an die Technologiestiftung Berlin. TU Berlin, Fachgebiet Theoretische Methoden der Bau- und Verkehrstechnik, Juli 1997.

[6] Kolbe, P.; Pfennigschmidt, S.; Pahl, P.J. : Verteiltes Facility Management in Telekommunikationsnetzen. Abschlußbericht des Fatima-Projektes, TU Berlin, Fachgebiet Theoretische Methoden der Bau- und Verkehrstechnik, Juni 1997.

[7] IAI Industrie Allianz für Interoperabilität :
Specifications Volume 1 bis 4.
Obermeyer Planen und Beraten, München 1996.

[8] Schneider, U.; Beucke, K. : Integration Concept based on a Communication Standard, IABSE-Symposium “Structures for the Future - The Search for Quality”,
Rio de Janeiro, 1999.

[9] Pahl, P.J.; Damrath, R. : Mathematische Grundlagen der Ingenieurinformatik, Kapitel 8.
Springer-Verlag, Berlin.

[10] Turau, V. : Algorithmische Graphentheorie. Addison-Wesley, Bonn, 1996, Seite 269.

[11] Firmenich, B.; Beucke , K. : A CAD-system design for multi-user requirements in civil engineering applications. Proceedings, 8th International Conference on Computing in Civil and Building Engineering, Stanford, August 2000.