Eine formale Algebra für die strukturierte Dokumentenverarbeitung

Ulrike Lucke
Universität Rostock, Institut für Informatik, Lehrstuhl für Rechnerarchitektur
ul@informatik.uni-rostock.de

Abstract: Aufgrund der zunehmenden Komplexität von Dokumenten bzw. Dokumentenbeschreibungssprachen – insbesondere im Kontext des eLearning – werden auch die damit verbundenen Prozesse der Erstellung, Verarbeitung und Nutzung von Dokumenten immer vielschichtiger. Dem werden existierende Werkzeuge und Architekturen nur noch bedingt gerecht. Ausgehend von der in diesem Beitrag beschriebenen Dokument-Algebra, die ein formales Datenmodell sowie darauf basierende Operationen umfasst, ist eine von Details konkreter Implementierungen abstrahierende Betrachtung der strukturierten Dokumentenverarbeitung ableitbar, aus der sich wiederum Konsequenzen für eine optimierte Gestaltung von Werkzeugen und Architekturen ergeben.

1 Motivation

Warum Dokumentenverarbeitung? Ein wesentlicher Anteil der eLearning-Aktivitäten der vergangenen Jahre hat sich vor allem mit der Produktion multimedialer Lehr- und Lernmaterialien befasst [DLR03]; die Erforschung grundlegender Konzepte sowie die Entwicklung allgemein gültiger Werkzeuge und Architekturen für das eLearning spielten bislang eine eher untergeordnete Rolle. Inzwischen ist eine zunehmende Verwendung von XML-basierten Markup-Sprachen zu verzeichnen [Rin03], die aufgrund ihrer flexiblen Aufbereitungsmöglichkeiten zwar die Wiederverwendbarkeit der Materialien und damit deren Wirtschaftlichkeit erhöhen, aber gleichzeitig die Erstellung und Verarbeitung der Dokumente erheblich komplizieren. Es werden daher generelle Konzepte, Verfahren und Hilfsmittel zur Strukturierung und damit Optimierung der mit derartigen Dokumenten verbundenen Prozesse benötigt [LNT05].

Generell kann das eLearning als ein treffendes Beispiel der vielfältigen Einsatzbereiche der rechnergestützten Dokumentenverarbeitung betrachtet werden. Hier liegt eine große Menge von Materialien in verschiedenster Form vor, die gemeinsam zum Einsatz kommen sollen. Im Unterschied zu anderen Anwendungen werden hier hohe Anforderungen an die Qualität der Dokumente nicht nur aus inhaltlicher und formeller, sondern auch aus didaktischer und gestalterischer Sicht gestellt. Dies führt zu einer großen Vielfalt von Dokumenttypen und Beschreibungssprachen sowie einem vielschichtigen Erstellungs- und Nutzungsprozess.

Warum eine Formalisierung? Ein fundiertes und umfassendes formales Modell wird häufig als zentrale Voraussetzung für den Entwurf leistungsfähiger Beschreibungsformen und Verarbeitungstechniken für Daten angesehen; zudem stellt es einen wichtigen Ansatzpunkt für Optimierungen dar [SM03]. In der Praxis führen formale Modelle leider selten

zu relevanten Implementierungen oder gar Re-Implementierungen bestehender Systeme, da ästetische Ideale der Theorie und pragmatische Forderungen der Anwendung oft weit auseinander klaffen [Sub95].

Formalismen können allerdings eine sachdienliche Vergleichsbasis für reale Systeme bilden, denn gerade hierfür wird eine von Details der Implementierung abstrahierende Sichtweise benötigt. Darüber hinaus können die aus einem formalen Modell abgeleiteten Schlussfolgerungen die grundlegenden Prinzipien für eine nachfolgende Implementierung von Werkzeugen bzw. Gestaltung von Architekturen nachhaltig beeinflussen [Kor04]. Ziel dieser Arbeit ist es daher, aus den bestehenden Formen der Dokumentenverarbeitung einen generellen Formalismus abzuleiten, der anschließend wiederum zur Verbesserung von Werkzeugen und Architekturen nicht nur des eLearning, sondern der strukturierten Verarbeitung von Dokumenten im Allgemeinen genutzt werden soll.

2 Definition einer Dokument-Algebra

2.1 Datentypen

In Anlehnung an den traditionellen Dokumentenbegriff als hierarchisch gegliedertes Printmedium (z. B. Artikel, Buch) sollen nachfolgend auch digitale Dokumente als Hierarchie modelliert werden. Durch Links ist eine *vernetzte Struktur* von Informationseinheiten (z. B. Hypertext) daraus konstruierbar, wie sie bereits 1945 durch Bush beschrieben wurde [Bus45], die dem aktuellen Aufbau digitaler Dokumente gerecht wird. Dabei ist zunächst zwischen einfachen und zusammengesetzten Bestandteilen zu unterscheiden.

Die in einem Dokument enthaltenen Medienobjekte stellen die für den Betrachter primär sichtbaren Bestandteile eines Dokuments dar. Sie können von verschiedenem Typ sein, z. B. Text, Bild, Animation, Video oder Sound. Die Menge dieser einfachen, je nach Dokument und Anwendungsgebiet variierenden Datentypen soll im Folgenden unter dem Datentyp *ATOMIC* zusammengefasst werden.

Definition 1: *Medien- oder atomares Objekt*

ein Objekt $x = name{:}value$ vom Datentyp *ATOMIC* wird als Medien- bzw. atomares Objekt bezeichnet. Dabei gibt *name*, gefolgt von einem Doppelpunkt, einen optionalen Bezeichner zur Identifikation des Objekts an, und *value* verweist auf oder enthält den Inhalt des Objekts.

Es sind i. A. keine Aussagen über eine interne Strukturierung dieser Objekte ersichtlich, bzw. die Details der logischen Interpretation oder physikalischen Speicherung von Daten sollen hier vernachlässigt werden. Die Objekte werden als atomar angenommen, wenn sie sich dem Betrachter als untrennbare Einheit präsentieren und von ihm ohne Kenntnis ggf. vorhandener interner Strukturen vollständig erfasst werden können.

Die Strukturierung in Dokumenten erfolgt durch Bildung von Beziehungen, die einzelne Objekte zu einem zusammengesetzten Objekt formen. Dieser Datentyp soll im Folgenden als *COMP* (für composite object) bezeichnet werden und wird rekursiv wie folgt definiert:

Definition 2: *Zusammengesetztes Objekt*

Ein Tupel $x = name{:}(y_1,\ldots,y_n)$ mit $y_1,\ldots,y_n \in (ATOMIC \cup COMP)$ $(n \in N^+)$ wird als *zusammengesetztes Objekt* vom Datentyp *COMP* bezeichnet. Dabei gibt *name*, gefolgt von einem Doppelpunkt, einen optionalen Bezeichner zur Identifikation des Objekts an.

Zusammengesetzte Objekte sind (u. U. mehrfach) verschachtelte Tupel. Sie können auch als Strukturobjekte bezeichnet werden, da sie den internen Aufbau bzw. die Struktur eines Dokuments bestimmen. Strukturobjekte werden durch den Rezipienten eines Dokuments nur sekundär wahrgenommen; sie äußern sich bspw. in der Anordnung oder Formatierung ihrer Bestandteile.

Die Gesamtheit aller atomaren und zusammengesetzten Objekte soll nachfolgend zur Vereinfachung nur noch als *Objekt* bezeichnet werden. Für das leere Objekt wird die Bezeichnung ε gewählt. Dabei gilt $(a, \varepsilon, b) = (a, b)$ sowie $(\varepsilon) = \varepsilon$.

Die in einem Dokument bzw. Objekte enthaltenen Bestandteile erhält man durch Dekomposition der zusammengesetzten Objekte.

Definition 3: *Bestandteile* von Objekten

Die Menge der *Bestandteile* eines zusammengesetzten Objekts $x = (y_1, \ldots, y_n)$ ist definiert als $constituents(x) = \{y_1, \ldots, y_n\}$.

Alle Objekte können attributiert, d. h. mit weiterführenden Informationen angereichert werden. Zwar werden Attribute ebenso wie atomare Medienobjekte in sich nicht weiter strukturiert, sie werden jedoch zur textuellen Beschreibung von für den Rezipienten nicht direkt sichtbaren Daten (z. B. gestalterische oder technische Parameter) genutzt.

Definition 4: *Attribute* von Objekten

Die Beschreibung von Eigenschaften eines Objekts x durch die Attribute $a_1, \ldots, a_n$ $(n \in N)$ wird ausgedrückt durch $x@(a_1Name{:}a_1,\ldots,a_nName{:}a_n)$, wobei a_iName die jeweiligen Bezeichner der Attribute a_i enthalten.

Die Angabe von Bezeichnern ist für Attribute zwingend, da sie die Attributwerte zu den zu spezifizierenden Eigenschaften des Objekts zuordnen. Bei Objekten mit nur einem Attribut kann die Klammerung der Attributmenge weggelassen werden.

Auf der Basis von Medien- und Strukturobjekten in Dokumenten sowie deren Attributen lassen sich in Anlehnung an Begriffe der Graphentheorie weitere Definitionen im Rahmen einer Dokument-Algebra ableiten:

- Neben Positionsnummern können auch die Bezeichner von Objekten und Attributen für Pfadangaben genutzt werden.
- Auf Basis von Pfaden sind Links in Dokumenten beschreibbar, die inkludierend (d. h. an Stelle der Linkquelle) oder referenzierend (d. h. in Ergänzung zur Linkquelle) interpretierbar sind.
- Unter den Bestandteilen von Dokumenten nehmen die Wurzel und der Saum (d. h. eine Verkettung aller enthaltenen Medienobjekte) eine besondere Stellung ein.
- Es können Relationen wie die Saum- und Strukturäquivalenz oder die Identität von Dokumenten definiert werden.
- Beschreibungssprachen für Dokumente führen von untypisierten Tupelmengen zu verschiedenen Klassen von Strukturobjekten mit variierenden Freiheitsgraden.

- Die Abbildung zwischen Dokumenten und Dokumentenbeschreibungssprachen erfolgt durch Definition von Validitätskriterien.

Aus Platzgründen wird hier jedoch auf eine vollständige Angabe dieser Definitionen verzichtet; für weiterführende Darstellungen sei z. B. auf [Fro02][FCD02] verwiesen.

2.2 Operationen

Die auf Dokumenten bzw. deren Objekten basierenden Operationen lassen sich anhand des Lebenszyklus' in die Komposition und Transformation von Dokumenten sowie deren Präsentation für bzw. Rezeption durch den Endnutzer unterteilen.

Komposition von Dokumenten. Der Erstellungsprozess besteht im Wesentlichen aus der hierarchischen Strukturierung von Dokumentfragmenten. Dabei können feingranulare Objekte *bottom-up* zu übergeordneten Strukturobjekten zusammengefasst werden, oder eine Grobstruktur wird *top-down* sukzessive durch Einfügen von Struktur- oder Medienobjekten verfeinert. Aus formaler Sicht sind derartige Operationen vorwiegend dyadisch, d. h. zwei Objekte werden (ggf. parametrisiert) in ein drittes Objekt umgeformt:

$$(COMP \cup ATOMIC) \times (COMP \cup ATOMIC) \rightarrow COMP$$

Mit der relationalen Algebra wurden schon 1970 durch Codd grundlegende Operationen beschrieben [Cod70], die allerdings auf einfachen, gleichförmigen Tupelmengen arbeiten. Bei der Verarbeitung hierarchischer Strukturen sind jedoch verschachtelte Objekte sowie die Ordnung der Bestandteile von Objekten zu beachten. Die Dokument-Algebra zielt darüber hinaus nicht auf die gleichzeitige Verarbeitung großer Datenmengen, sondern auf eine abstrakte Beschreibung der Verarbeitung einzelner Dokumente. Daher sind die Operationen in diesem Zusammenhang wie folgt neu zu definieren bzw. zu ergänzen:

Definition 5: *Vereinigung* von Objekten

Die Vereinigung (union) $\cup$ zweier Objekte $a = (a_1,\ldots,a_n)$ und $b = (b_1,\ldots,b_m)$ enthält die Bestandteile beider Objekte mit Ausnahme der auch in a vorkommenden Bestandteile von b:

$$a \cup b = \begin{cases} \varepsilon & \text{falls } n = 0 \text{ und } m = 0 \\ (a_1,\ldots,a_n,b_1', \ldots, b_m') & \text{sonst} \end{cases}$$

$$\text{wobei} \qquad b_j' = \begin{cases} \varepsilon & \text{falls } b_j \in constituents(a) \\ b_j & \text{sonst} \end{cases}$$

Definition 6: *Durchschnitt* von Objekten

Der Durchschnitt (intersection) $\cap$ zweier Objekte $a = (a_1,\ldots,a_n)$ und $b = (b_1,\ldots,b_m)$ enthält die in beiden Objekten vorkommenden Bestandteile:

$$a \cap b = \begin{cases} \varepsilon & \text{falls } n = 0 \text{ oder } m = 0 \\ (a_1',\ldots,a_n') & \text{sonst} \end{cases}$$

$$\text{wobei} \qquad a_j' = \begin{cases} a_j & \text{falls } a_j \in constituents(b) \\ \varepsilon & \text{sonst} \end{cases}$$

Definition 7: *Differenz* von Objekten

Die Differenz (difference) – zweier Objekte $a = (a_1,\ldots,a_n)$ und $b = (b_1,\ldots,b_m)$ enthält die im ersten, aber nicht im zweiten Objekt vorkommenden Bestandteile:

$$a - b = \begin{cases} \varepsilon & \text{falls } n = 0 \\ (a_1',\ldots,a_n') & \text{sonst} \end{cases}$$

$$\text{wobei} \qquad a_j' = \begin{cases} a_j & \text{falls } a_j \notin constituents(b) \\ \varepsilon & \text{sonst} \end{cases}$$

Definition 8: *Kreuzprodukt* von Objekten

Das Kreuzprodukt (cartesian product) $\times$ zweier Objekte $a = (a_1,\ldots,a_n)$ und $b = (b_1,\ldots,b_m)$ enthält jeden Bestandteil a_i aus a mit jedem Bestandteil b_j aus b zu einem Bestandteil (a_i, b_j) des Zielobjekts verknüpft:

$$a \times b = \begin{cases} \varepsilon & \text{falls } n = 0 \text{ oder } m = 0 \\ ((a_1, b_1),(a_1, b_2), \ldots, (a_1, b_m), (a_2, b_1), \ldots, (a_n, b_m)) & \text{sonst} \end{cases}$$

Definition 9: *Verbund* von Objekten

Der Verbund (join) $\bowtie$ zweier Objekte $a = (a_1,\ldots,a_n)$ und $b = (b_1,\ldots,b_m)$ bzgl. des Prädikats $p{:}(COMP \cup ATOMIC) \times (COMP \cup ATOMIC) \rightarrow BOOL$ enthält diejenigen Bestandteile a_i aus a mit denjenigen Bestandteilen b_j aus b zu jeweils einem Bestandteil (a_i, b_j) des Zielobjekts verknüpft, für die $p(a_i, b_j)$ wahr ist:

$$a \bowtie |_p \; b = \begin{cases} \varepsilon & \text{falls } n = 0 \text{ oder } m = 0 \\ ((a_1, b_1)',\ldots, (a_1, b_m)', (a_2, b_1)', \ldots, (a_n, b_m)') & \text{sonst} \end{cases}$$

$$\text{wobei} \qquad (a_i, b_j)' = \begin{cases} (a_i, b_j) & \text{falls } p(a_i, b_j) = true \\ \varepsilon & \text{sonst} \end{cases}$$

Definition 10: *Konkatenation* von Objekten

Die Konkatenation (concat) $\bullet$ zweier Objekte $a = (a_1,\ldots,a_n)$ und $b = (b_1,\ldots,b_m)$ enthält die Aufreihung aller Bestandteile beider Objekte:
$a \bullet b = (a_1, \ldots, a_n , b_1, \ldots, b_m)$

Definition 11: *Paarung* von Objekten

Die Paarung (pair) + zweier Objekte $a = (a_1,\ldots,a_n)$ und $b = (b_1,\ldots,b_m)$ enthält die abwechselnde Aufreihung je eines Bestandteils der beiden Objekte:

$$a + b = \begin{cases} \varepsilon & \text{falls } n = 0 \text{ oder } m = 0 \\ (a_1, b_1,\ldots, a_k, b_k) & \text{sonst} \end{cases}$$

$$\text{wobei} \qquad k = min(n, m)$$

Definition 12: *Einfügung* eines Objekts

Die Einfügung (insert) λ erweitert ein zusammengesetztes Objekt $a = (a_1,\ldots,a_n)$ um einen neuen Bestandteil b nach dem *i*-ten Bestandteil ($i \in N$, $0 \le i \le n$):

$$a\lambda|_i b = \begin{cases} b & \text{falls } n = 0 \\ (a_1,\ldots,a_i,b,a_{i+1},\ldots,a_n) & \text{sonst} \end{cases}$$

Für eine vollständige Definition der Operationen sind auch die den Operanden zugrunde liegenden Schemata bzw. Beschreibungssprachen zu beachten. Da diese allerdings bereits in Abschnitt 2.1 unberücksichtigt blieben, soll auch hier auf die Angabe von Schema-Bedingungen verzichtet werden. Außerdem wird die Identifikation von Bestandteilen auf Positionsnummern beschränkt, da Pfadangaben nicht definiert wurden.

Transformation von Dokumenten. Bei der Umformung bestehender Objekte bzw. Dokumente kommen vorwiegend monadische Operationen zur Anwendung, die ein zusammengesetztes Objekt (ggf. parametrisiert) in ein anderes Objekt umwandeln:
$COMP \rightarrow (COMP \cup ATOMIC)$

Wie bereits für die Komposition kann auch für die Transformation von Objekten ansatzweise auf die relationale Algebra [Cod70] zurückgegriffen werden, der die ersten der nachfolgend definierten Operationen entstammen. Weitere Operationen für verschachtelte Tupel-Sequenzen wurden z. B. durch Güting et al. konzipiert [GZC89] und sind hier für die Verarbeitung von hierarchisch strukturierten Dokumenten angepasst.

Definition 13: *Projektion* auf ein Objekt

Die Projektion π wählt aus einem zusammengesetzten Objekt $a = (a_1,\ldots,a_n)$ dessen durch i_1 bis i_p (alle $i \in N$, $0 \le i \le n$) identifizierte Bestandteile aus:

$$a\pi|_{i_1,\ldots,i_p} = \begin{cases} \varepsilon & \text{falls } n = 0 \\ (a_{i_1},\ldots,a_{i_p}) & \text{sonst} \end{cases}$$

Definition 14: *Selektion* aus einem Objekt

Die Selektion σ wählt aus einem zusammengesetzten Objekt $a = (a_1,\ldots,a_n)$ diejenigen Bestandteile aus, für die das Prädikat p: $(COMP \cup ATOMIC) \rightarrow BOOL$ wahr ist:

$$a\sigma|_p = \begin{cases} \varepsilon & \text{falls } n = 0 \\ (a_1',\ldots, a_n') & \text{sonst} \end{cases}$$

$$\text{wobei} \qquad a_j' = \begin{cases} a_j & \text{falls } p(a_j) = true \\ \varepsilon & \text{sonst} \end{cases}$$

Wie auch bei Datenbank-Algebren wird hier zwischen der Auswahl von Objekten nach ihrer Bezeichnung (Projektion) oder nach ihrem Inhalt (Selektion) unterschieden. Beides lässt sich im Kontext einer Dokument-Algebra jedoch durch Definition eines geeigneten Prädikats zu einer einzigen Operation zusammenfassen, da die Operation nicht auf eine Tupelmenge sondern lediglich ein (ggf. verschachteltes) Tupel angewandt wird.

Definition 15: *Umbenennung* eines Objekts

Die Umbenennung (rename) ρ in einem zusammengesetzten Objekt $a = (a_1,\ldots,a_n)$ ändert den Namen des Bestandteils N in N':

$$a\rho|_{N,N'} = \begin{cases} \varepsilon & \text{falls } n = 0 \\ (a_1',\ldots,a_n') & \text{sonst} \end{cases}$$

$$\text{wobei} \qquad a_j' = \begin{cases} N':x & \text{falls } name = N \\ & \text{mit } a_j = name:x \\ a_j & \text{sonst} \end{cases}$$

Definition 16: *Entfernung* von Teilobjekten

Die Entfernung (delete) δ löscht alle Bestandteile aus einem zusammengesetzten Objekt $a = (a_1,\ldots,a_n)$, für die das Prädikat p: $(COMP \cup ATOMIC) \rightarrow BOOL$ wahr ist:

$$a\delta|_p = \begin{cases} \varepsilon & \text{falls } n = 0 \\ (a_1',\ldots,a_n') & \text{sonst} \end{cases}$$

$$\text{wobei} \qquad a_j' = \begin{cases} \varepsilon & \text{falls } p(a_j) = true \\ a_j & \text{sonst} \end{cases}$$

Definition 17: *Entfernung von Duplikaten* unter Teilobjekten

Die Duplikatentfernung (trim) τ entfernt alle mehrfach vorkommenden Bestandteile eines zusammengesetzten Objekts $a = (a_1,\ldots,a_n)$:

$$a\tau = \begin{cases} \varepsilon & \text{falls } n = 0 \\ (a_1',\ldots,a_n') & \text{sonst} \end{cases}$$

$$\text{wobei} \qquad a_j' = \begin{cases} \varepsilon & \text{falls } \exists k \in N, 1 \le k < j \text{ mit } a_j = a_k \\ a_j & \text{sonst} \end{cases}$$

Definition 18: *Gruppierung* von Teilobjekten

Die Gruppierung (nest) ν entfernt aus einem zusammengesetzten Objekt $a = (a_1,\ldots,a_n)$ die durch $i_1, \ldots, i_p$ (alle $i \in N$, $0 \le i \le n$) identifizierten Bestandteile, fasst sie zu einem neuen Objekt zusammen und fügt dies anstelle von a_{i_1} in a ein:

$$a\nu|_{i_1,\ldots,i_p} = \begin{cases} \varepsilon & \text{falls } n = 0 \\ (a_1',\ldots,a_n') & \text{sonst} \end{cases}$$

$$\text{wobei} \qquad a_j' = \begin{cases} (a_{i_1},\ldots,a_{i_p}) & \text{falls } j = i_1 \\ \varepsilon & \text{falls } j \in (i_2, \ldots, i_p) \\ a_j & \text{sonst} \end{cases}$$

Definition 19: *Verteilung* von Teilobjekten

Die Verteilung (unnest) μ entfernt den durch i ($i \in N,\ 1 \le i \le n$) identifizierten Bestandteil aus einem zusammengesetzten Objekt $a = (a_1,\ldots,a_n)$ und fügt an dieser Stelle dessen Bestandteile direkt in a ein:

$$a\mu|_i = \begin{cases} \varepsilon & \text{falls } n = 0 \\ (a_1',\ldots,a_n') & \text{sonst} \end{cases}$$

$$\text{wobei} \qquad a_j' = \begin{cases} a_{i_1}, \ldots, a_{i_m} & \text{falls } j = i \text{ mit } a_i = (a_{i_1}, \ldots, a_{i_m}) \\ a_j & \text{sonst} \end{cases}$$

Bei der Transformation von Dokumenten bzw. Objekten können auch alle beschriebenen dyadischen Operationen zum Einsatz kommen. Darüber hinaus existiert eine Reihe von weiteren Funktionen (z. B. *count*, *avg*, *sum*, *min*, *max*, *getPosition*, *isempty*, *exists*) als Submenge der monadischen Operationen, die zusammengesetzte auf atomare Objekte (numerische oder Boolesche Werte) abbilden: $COMP \rightarrow ATOMIC$. Diese Funktionen sind wie üblich definiert und daher hier nicht explizit angeführt.
Auch wurde hier zum Zwecke der Übersichtlichkeit die Manipulation von Attributen und Links nicht betrachtet. Durch eine Verallgemeinerung von Medienobjekten, Attributen und Links zu einem abstrakten Objekttyp mit angepasster Notation [BMR99][GHW99] sind jedoch die Operationen der Dokument-Algebra mit geringfügigen Erweiterungen von Medien- und Strukturobjekten auch auf Attribute und Links anwendbar.

Präsentation und Rezeption. Auf weitere Phasen im Dokument-Lebenszyklus soll an dieser Stelle nicht näher eingegangen werden, da sich die dazu benötigten Operationen auf die bereits definierten zurückführen lassen. So stellt die *Publikation* eines Dokuments eine Transformation in eine andere Beschreibungssprache dar, wofür die Grundoperationen in einem (häufig komplexen) Algorithmus zu kombinieren sind. Ein typisches Beispiel dafür ist die Generierung von PDF oder HTML aus XML-Dialekten. Zur Präsentation von Dokument(kollektion)en wird darüber hinaus eine *Navigation*, d. h. eine selektive Publikation unter Auswertung interaktiver Pfadangaben, benötigt. Eine weitere Form der Interaktion mit Dokumenten ist deren *Annotation*, was wiederum eine Erweiterung um neue Objekte und damit eine (einfache) Re-Komposition des Dokuments darstellt. Ferner liegen auch Metadaten zu Dokumenten und ihren Bestandteilen sowie semi-automatische Verfahren zur Verknüpfung von Dokumenten im Fokus der Formalisierung.

2.3 Schlussfolgerungen

Da die Dokument-Algebra im Rahmen dieses Beitrags nicht vollständig wiedergegeben werden kann und einige der aus ihr ableitbaren Schlussfolgerungen daher nur vage zu motivieren sind, sollen nachfolgend lediglich zentrale Anforderungen an Werkzeuge und Architekturen überblicksartig angegeben werden.

Autorenwerkzeuge unterstützen die Komposition von Dokumenten aus atomaren und zusammengesetzten Objekten entsprechend der zugrunde gelegten Beschreibungssprache. Hinsichtlich ihrer *Funktionalität* sollten sie daher:

- eine hinreichende Menge atomaren Daten- bzw. Medientypen (ob vollständig integriert oder als vereinfachte Vorschau) unterstützen,
- alle in Abschnitt 2.2 definierten grundlegenden Operationen zur Komposition, aber auch zur Transformation von Objekten bzw. Dokumenten umsetzen,
- durch eine ständige Validitätsprüfung auf alle der Sprachdefinition widersprechenden Objekte (an der falschen Stelle, vom falschen Typ oder fehlend) hinweisen,
- den Dokumentinhalt (d. h. sowohl Medien- und Strukturobjekte als auch Attribute und Assoziationen) adäquat visualisieren sowie
- das Konzept der Modularisierung und Wiederverwendung von Dokumenten bzw. deren Bestandteilen auf verschiedenen Granularitätsebenen unterstützen.

Weitere wichtige Aspekte bei der Werkzeuggestaltung sind neben einer komfortablen und intuitiven Benutzung (*Usability*) vor allem die Möglichkeiten zur *Integration* in eine heterogene Umgebung bzw. zur Einbettung des Dokuments in externe Prozesse und Infrastrukturen. Die Werkzeuge sollten geeignete Schnittstellen aufweisen, über die ihre Interoperabilität mit anderen Systemen gewährleistet ist.

Transformationen sollten wahlweise als selbständiges Programm gestartet oder als Routine in anderen Werkzeugen aufgerufen werden können; die Algorithmen müssen daher in einer flexiblen Beschreibung vorliegen.

- Transformationen zur visuell orientierten Formatierung eines Dokuments während des Authorings (Voransicht) sollten direkt in das Autorenwerkzeug eingebettet sein. Sie werden durch den Autor individuell parametrisiert und sind daher eng an den Erstellungskontext gebunden.
- Transformationen zur Dokumentenverwaltung und -nutzung (sprachintern, in andere Beschreibungssprachen oder zur Generierung eines Layouts) müssen frei aufrufbar sein, da sie mit einem spezifischen Anwendungskontext verbunden sind.

Diese Unterscheidung wird darüber hinaus durch die Komplexität der Algorithmen unterstützt. Während Formatierungen einzelner, vom Autor editierter Objekte vglw. schnell angewandt werden können und daher gut in das Autorenwerkzeug integrierbar sind, erfordern strukturelle Umwandlungen von komplexen Dokumenten einen bedeutend höheren Aufwand und sollten daher durch optimierte Spezialwerkzeuge vorgenommen werden. Dabei bietet sich eine Auslagerung auf zentrale Server an, was wiederum durch eine geeignete Architektur unterstützt werden muss.

Architekturen für die verteilte Dokumentenverarbeitung werden nicht erst im Zuge der vielfältigen, häufig internet-basierten Anwendungsszenarien benötigt. Insbesondere im eLearning führen hohe Qualitätsanforderungen bereits bei der Erstellung von Dokumenten zu einem großen, interdisziplinären und häufig verteilten Autorenteam [Kor04]. Eine hochgradig modulare Architektur sollte deshalb alle Phasen des Dokumentlebenszyklus in der vom jeweiligen Nutzer präferierten Ausprägung unterstützen. Die Unabhängigkeit von spezifischen Plattformen, aber auch von konkreten Werkzeugen oder einfach nur von Bearbeitungsmodi spielt dabei eine zentrale Rolle. Ein innovativer Ansatz hierfür ist die freie, dienste-basierte Komposition der Umgebung durch den Nutzer entsprechend seinen individuellen Anforderungen [LT04], bspw. unter Verwendung von Web Services.

3 Verwandte Arbeiten

Bestehende Arbeiten zur Definition oder Präzisierung des Dokumentenbegriffs lassen sich der physikalischen, formalen oder konzeptuellen Ebene zuordnen [SM03], wobei in diesem Zusammenhang lediglich formale Ansätze betrachtet werden sollen. Dazu sind in der Vergangenheit zahlreiche Arbeiten entstanden, die sich zunächst auf eine syntaktische Beschreibung der *Datenstrukturen* konzentrierten, wobei v. a. hypertextuelle und (semi-) strukturierte Dokumentstrukturen betrachtet wurden:

- Das Dexter-Modell für Hypertexte [HS94] ist aus proprietären Hypertextformaten entstanden und bildete den Ausgangspunkt für eine formale Definition sowie erste Ansätze der Implementierung der Hypertext Markup Language (HTML).
- Eine algebraische Spezifikation der Struktur von Hypertexten sowie der Navigation darin wurde u. a. durch Fronk modelliert [Fro02]. Der Fokus lag dort auf konkreten Dokument-Instanzen (ohne Beachtung von Beschreibungssprachen).
- Verschiedene formale Modelle beschäftigen sich mit syntaktischen und semantischen Aspekten von XML-basierten Dokumenten, wie z. B. [MLM01].

Die Ansätze zu einer Formalisierung der *Verarbeitung* (semi-)strukturierter Daten entstammen v. a. der Datenbanktheorie. Sie bilden dort die Grundlage einer formalen Definition von Anfragesprachen.

- Die relationale Algebra [Cod70] definiert Operationen auf einfachen, unsortierten Tupelmengen mit lediglich atomaren Elementen.
- Die Manipulation von verschachtelten Tupelmengen wurde in verschiedenen NST-Algebren formalisiert [BT99][GZC89].
- In objektorientierten Algebren [LV96] werden neben der Strukturierung von Objekten zusätzlich Aspekte wie Vererbung oder Objekt-Identität betrachtet.
- Mechanismen wie XQuery oder XPath [W3C] erlauben die Suche bzw. Lokalisation in semi-strukturierten Dokumenten, nicht jedoch deren Erstellung oder Umformung.

Die genannten Verfahren wurden für den vorliegenden Beitrag adaptiert, erweitert und zu einer Dokument-Algebra zusammengefasst. Etablierte Mechanismen von Datenbank-Algebren wurden auf moderne Formen semi-strukturierter Dokumente angewandt. Auf rekursive Operatoren wurde dabei verzichtet, da diese aufgrund hoher Freiheitsgrade der Dokumentstrukturen nicht ohne Weiteres einsetzbar sind.

4 Zusammenfassung und Ausblick

Auf Basis existierender Arbeiten zur Formalisierung des Aufbaus und der Verarbeitung von Datenmengen wurde eine Algebra für semi-strukturierte Dokumente beschrieben. Atomare Medien- und zusammengesetzte Strukturobjekte bilden die Datentypen der Algebra, auf denen 15 grundlegende Operationen zur Komposition und Transformation von Dokumenten definiert werden. Im Unterschied zu vergleichbaren Ansätzen können alle Objekte attributiert und mit Bezeichnern versehen werden, was die Definition von Pfaden und damit auch von Links in bzw. zwischen Dokumenten vereinfacht. Durch eine Einbettung der Operationen in eine algorithmische Beschreibung lässt sich somit der vollständige Lebenszyklus eines Dokuments auf einem abstrakten Niveau modellieren.

Zu dem im Rahmen dieses Beitrags nicht skizzierten Funktionsumfang der Algebra zählen u. a. eine formale Definition ausgewählter *Eigenschaften* von Dokumenten, die *Mehrdimensionalität* von kontext-behafteten Dokumenten [LTV03][SGR00] sowie die *Validität* von Dokumenten bezüglich einer *Beschreibungssprache*.

Aktuelle Arbeiten umfassen die exemplarische *Verifikation* der entwickelten Algebra anhand der XML-basierten Beschreibungssprache <ML>³ [LTV03] und der darin implementierten, umfangreichen Dokumentkollektionen aus dem Anwendungsgebiet eLearning [WWR]. Die algebraische Formalisierung wird darüber hinaus derzeit für einen umfassenden *Vergleich* bestehender Systeme zur Dokumentenverarbeitung auf abstrakter Ebene genutzt. Im Resultat werden die an effektive *Werkzeuge* und *Architekturen* zu stellenden Anforderungen abgeleitet und durch eine prototypische Implementierung untermauert. Die Ergebnisse dieser Arbeiten entstammen zwar dem eLearning, sind jedoch nicht auf dieses Anwendungsgebiet beschränkt; für die strukturierte Verarbeitung digitaler Dokumente im Allgemeinen wird hier ein großes Potenzial gesehen.

5 Literatur

[BMR99] D. Beech, A. Malhotra, M. Rys: „A Formal Data model and Algebra for XML“, Submission to the W3C, Oktober 1999.

[BT99] C. Beeri, Y. Tzaban: „SAL: An Algebra for Semistructured Data and XML“, ACM SIGMOD Workshop on The Web and Databases (WebDB), 1999.

[Bus45] V. Bush: „As we may think“. The Atlantic Monthly 176/1, Juli 1945.

[Cod70] E. F. Codd: „A Relational Model of Data for Large Shared Data Banks“, ACM Communications, Vol. 13, Nr. 6, Juni 1970.

[DLR03] DLR-Projektträger NMB: „Kursbuch eLearning“, Bonn / St. Augustin, 2003.

[FCD02] L. Feng, E. Chang, T. Dillon: „A Semantic Network-Based Design Methodology for XML Documents“, ACM Transactions on Information Systems, Vol. 20, Nr. 4, October 2002.

[Fro02] A. Fronk: „Algebraische Semantik einer objektorientierten Sprache zur Spezifikation von Hyperdokumenten“, Dissertation, Universität Dortmund, 2002.

[GHW99] R. Goldman, J. McHugh, J. Widom: „From Semistructured Data to XML: Migrating the Lore Data Model and Query Language“, ACM SIGMOD Workshop on The Web and Databases (WebDB), 1999.

[GZC89] R. H. Güting, R. Zicari, D. M. Choy: „An Algebra for Structured Office Documents“, ACM Communications on Office Information Systems, Vol. 7, Nr. 2, April 1989.

[HS94] F. Halasz, M. Schwartz: „The Dexter Hypertext Reference Model“, ACM Communications, Vol. 37, Nr. 2, Februar 1994.

[Kor04] L. Kornelsen, U. Lucke, D. Tavangarian, M. Waldhauer, N. Ossipova: „Strategien und Werkzeuge zur Erstellung multimedialer Lehr- und Lernmaterialien auf Basis von XML“. 2. Deutsche e-Learning Fachtagung Informatik (DeLFI 2004), Paderborn, 2004.

[LV96] G. Lausen, G. Vossen: „Objekt-orientierte Datenbanken: Modelle und Sprachen“. Oldenbourg Verlag, 1996.

[LTV03] U. Lucke, D. Tavangarian, D. Voigt: „Multidimensional Educational Multimedia with <ML>³“. World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education (E-Learn), Phoenix/Arizona/ USA, 2003.

[LT04] U. Lucke, D. Tavangarian: „Why and How Education Becomes a Service: An eLearning Architecture Based on Web Services“. World Conference on Educational Multimedia, Hypermedia, and Telecommunications (Ed-Media), Lugano/Schweiz, 2004.

[LNT05] U. Lucke, K. Nölting, D. Tavangarian: „Strukturierte Lehr- und Lernmodelle für die Ausbildung im Fachgebiet Technische Informatik“. Information Technology (it), Vol. 47, Nr. 3, 2005.

[MLM01] M. Mani, D. Lee, R. R. Muntz: „Semantic Data Modeling using XML Schemas“, 20th International Conference on Conceptual Modeling, Yokohama/ Japan, LNCS 2224, Springer Verlag, Berlin, 2001.

[Rin03] U. Rinn, K. Bett, J. Wedekind, P. Zentel, D. M. Meister, W.F. Hesse: „Virtuelle Lehre an deutschen Hochschulen im Verbund“, Institut für Wissensmedien, Tübingen, 2003.

[SM03] A. Sengupta, S. Mohan: „Formal and conceptual models for XML structures – the past, present, and future“, Indiana University, Technical Report TR 137-1, April 2003.

[SGR00] Y. Stavrakas, M. Gergatsoulis, P. Rondogiannis: „Multidimensional XML“, 3rd International Workshop on Distributed Computing on the Web (DCW), LNCS 1830, Springer Verlag, Berlin, 2000.

[Sub95] K. Subieta, Y. Kambayashi, J. Leszczylowski, K. Yokota: „Object Algebras: What is Wrong?“ (unpublished), 1995.
http://citeseer.ist.psu.edu/42952.html

[WWR] Wissenswerkstatt Rechensysteme. http://www.wwr-project.de/

[W3C] World Wide Web Consortium. http://www.w3.org/