

Multimedia-Datenbanken

Kapitel 8: Hypermedia

Friedrich-Alexander-Universität Erlangen-Nürnberg
Technische Fakultät, Institut für Informatik
Lehrstuhl für Informatik 6 (Datenbanksysteme)

Prof. Dr. Klaus Meyer-Wegener

Wintersemester 2002 / 2003

Technische Universität Kaiserslautern
Fachbereich Informatik
AG Datenbanken und Informationssysteme

Dr. Ulrich Marder

Wintersemester 2003 / 2004

8. Hypermedia

- Erweiterung von Hypertext
- Knoten
 - jetzt auch mit Graphik, Photos, akustischen Aufzeichnungen und Videos
- "dynamische" Medien
 - nur *indirekt* durch Platzhalter-Knoten darstellbar
 - zeigt ein Symbol oder Ikone für die Aufnahme
 - Lautsprecher, Fernsehbildschirm, ...
 - Kurzbeschreibung als Text:
 - "Anmerkung von xyz zum Thema ", "Film über die Geschichte der "
 - Angabe über Dauer!
 - Tastenfeld:



- Hauptaugenmerk derzeit aber auf passiven visuellen Medien
 - Grund wohl auch: geringere Anforderungen an Hardware

Hypermedia (2)

die wichtigsten Systeme:

- **Intermedia**
 - Brown University (Providence, Rhode Island)
 - Unterstützung von Lehre und Forschung in einer Universitätsumgebung
- **NoteCards**
 - Xerox PARC
 - "Zettelkasten" für Recherche und Schreiben von Berichten
- **Neptune / HAM**
 - Tektronix
 - Frontend – Backend, Transaktionskonzept
- **Hyperties**
 - University of Maryland
 - IBM PC, auch ohne Maus
 - Ausbildung und Lehre, "Kioske", Museen

Hypermedia (3)

die wichtigsten Systeme (Forts.):

- **KMS**
 - Knowledge Systems
 - Nachfolger von ZOG (Carnegie-Mellon University)
 - keine Fenster! ein bis zwei Knoten auf dem Schirm
- **HyperCard**
 - Apple
 - Graphiken und Rasterbilder auf Karten direkt unterstützt
 - Tonaufnahmen als "Ressourcen", d. h. separate Datei
 - Abspielen mit HyperTalk-Kommando "play dateiname"

8.1 Das Dexter-Referenzmodell

[Hala94a]

□ Herkunft und Name:

- Oktober 1988, Dexter Inn in Sunapee, New Hampshire, USA: Workshop, organisiert von John Leggett und Jan Walker
- Gruppe von Hypermedia-Systementwerfern, von denen jeder umfangreiche Erfahrung mit der Entwicklung zumindest eines bekannten Hypermedia-Systems haben musste

□ Ziel:

- Konsens finden über Terminologie und Semantik der elementaren Hypermedia-Konzepte

□ weitere Treffen der "Dexter-Gruppe":

- Entwicklung eines Daten- und Prozessmodells als Referenz für neue Hypertextsysteme
- weitere Ausarbeitung und Formalisierung durch zwei Mitglieder, Frank Halasz und Mayer Schwartz, in einem Beitrag zum NIST-Workshop über Hypertext Standardization im Januar 1990

Dexter-Referenzmodell (2)

□ Zweck des Dexter-Referenzmodells:

- Vergleich von Systemen
- Austausch und Interoperabilität

□ drei Schichten:

- **Speicherungsschicht** (storage layer):
Netz von Knoten und Links
- **Laufzeitschicht** (run-time layer):
Interaktion der Benutzer mit dem System
- **Komponenteninhaltsschicht** (within-component layer):
Inhalt und Struktur der Knoten
- Speicherungsschicht dabei im Mittelpunkt,
einschl. Mechanismen zur Spezifikation von Anker und Präsentationen,
die die Schnittstellen zu den beiden anderen Schichten bilden

Dexter-Referenzmodell (3)

□ Speicherungsschicht:

- Datenbasis, bestehend aus (atomaren) **Komponenten**, die über **Links** verbunden sind
- Komponenten entsprechen Knoten
 - bewusst andere Bezeichnung, damit systemneutral
- generische Datenbehälter, innere Struktur nicht berücksichtigt

□ Komponenteninhaltsschicht:

- im Dexter-Modell nicht weiter spezifiziert, kann alles sein
- andere Referenzmodelle einsetzen! und mit Dexter kombinieren, z. B. ODA, IGES (www.nist.gov/iges/) usw.
- allerdings Schnittstelle zwischen Hypertext-Netz und Komponenteninhalt erforderlich: Adressierung von Stellen und Elementen *innerhalb* der Komponenten – **Verankerung** (anchoring)
 - bei Systemen wie Intermedia: Links zwischen *Teilen* von Knoten, nicht nur beim Ausgangspunkt, sondern auch beim Ziel

Dexter-Referenzmodell (4)

□ Laufzeitschicht:

- Hypermedia-Systeme nicht nur passive Strukturen, sondern auch Werkzeuge für Zugriff, Ansehen, Verändern
- im Dexter-Modell ebenfalls nicht weiter spezifiziert, nur ganz elementar
- allerdings auch hier die Schnittstelle beachten: **Präsentationsspezifikationen**
- Information darüber, wie Komponenten und Netze darzustellen sind, in die Speicherungsschicht aufnehmen
- Darstellung dann nicht nur vom Werkzeug bestimmt, sondern auch von der Komponente und/oder dem Link, der zu ihr geführt hat
 - Beispiel: Verweis auf Animation in Lernumgebung
 - Start eines *Viewers* für den normalen Benutzer
 - Start eines *Editors* für den Autor/Lehrer

Einfaches Speicherungsschicht-Modell

- **Hypertext =**
 - endliche Menge von Komponenten
 - und zwei Funktionen: **Auflöser** (resolver) und **Zugreifer** (accessor) für das Wiederauffinden von Komponenten
 - Abbildung einer Spezifikation der Komponente auf die Komponente selbst
- **Komponenten:**
 - **atomare** Komponenten:
 - in den meisten Systemen Knoten
 - **Links:**
 - Beziehungen zwischen anderen Komponenten
 - Folge von zwei oder mehr "Endpunkt-Spezifikationen", die auf Teile von Komponenten verweisen
 - **zusammengesetzte** Komponenten:
 - enthalten andere Komponenten
 - Struktur dabei gerichteter azyklischer Graph, d. h. gemeinsame Komponenten, aber nicht sich selbst wieder als Komponente

Speicherungsschicht-Modell (2)

- **global eindeutige Identifikation: unique identifier (UID)**
 - über einzelnen Hypertext hinaus
 - **Zugreifer**-Funktion muss zu jeder UID die zugehörige Komponente liefern können
- **UIDs Basis der Adressierung, aber allein nicht ausreichend:**
 - Verweise auf andere Komponenten auch über deren Eigenschaften, z.B. auf Text, der bestimmtes Wort enthält – liefert evtl. gar keine oder mehrere Komponenten
 - dafür **Komponentenspezifikation** in den Links:
müssen von der **Auflöser**-Funktion auf UIDs abgebildet werden
 - UID selbst auch zulässig, Auflöser dann Identität
- **Links zwischen Teilen von Komponenten:**
 - UID allein genügt nicht
→ **Anker** (anchor),
besteht aus ID und Wert
 - ID eindeutig innerhalb der Komponente
 - Wert spezifiziert auf beliebige Weise Stelle, Region, Eintrag usw. in einer Komponente, wird nur von der Anwendung interpretiert und kann sich ändern

Speicherungsschicht-Modell (3)

□ Spezifizierer

- Anker-ID mit Komponentenspezifikation verbinden
- enthält zusätzlich noch:
 - Richtung: FROM, TO, BIDIRECT, NONE
 - NONE wird verwendet, wenn Anker eigentlich kein Teil der Komponente, sondern Programm oder Skript, wie z. B. bei Tasten in HyperCard
 - Präsentationsspezifikation (s. unten)

□ Link dann:

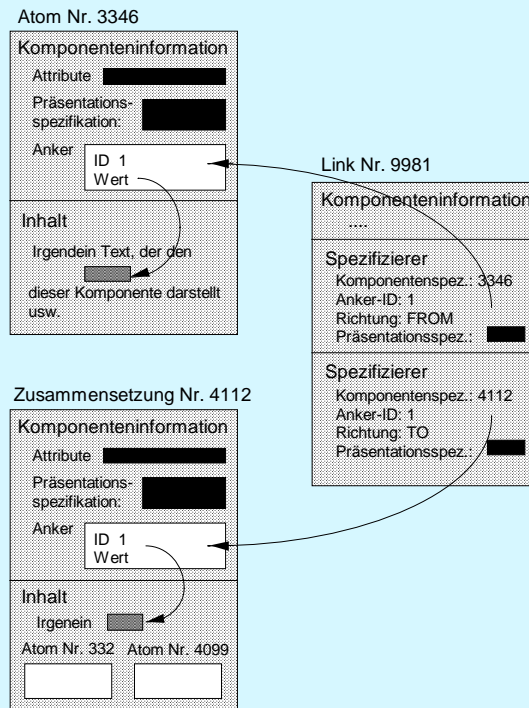
- Folge von zwei oder mehr Spezifizierern
- sehr allgemein: auch **Mehrweg-Links**
- einzige Bedingung:
mindestens ein Spezifizierer mit Richtung TO oder BIDIRECT

Speicherungsschicht-Modell (4)

□ Komponenten

- dargestellt durch Inhalt
- und **Komponenteninformation**:
 - Folge von Ankern
 - Präsentationsspezifikation
 - Menge von Attribut-Wert-Paaren
- Attribute beliebig,
z. B. Schlagwort (darf sich wiederholen!) oder Typ

Speicherungsschicht-Modell (5)



Speicherungsschicht-Modell (6)

□ Operationen:

- Hinzufügen und Löschen einer Komponente
- Ändern des Werts von Ankern oder Attributen
- Abruf einer Komponente über UID oder Spezifizierer
- **LinksTo:** bildet UID einer Komponente auf UIDs aller Links ab, die diese Komponente spezifizieren
- **LinksToAnchor:** bildet einen Anker auf die UIDs aller Links ab, die diesen Anker spezifizieren

□ Integritätsbedingungen (Invarianten):

- Zugreifer-Funktion (UID → Komponente) muss umkehrbar sein, d. h. alle Komponenten haben eine UID
- Auflöser-Funktion muss potenziell alle gültigen UIDs liefern können
- keine Zyklen in der Zusammensetzungsstruktur, d. h. keine Komponente enthält sich selbst
- Link-Konsistenz: Komponentenspezifikation muss auf eine existierende Komponente führen (d. h. referenzielle Integrität)

Einfaches Laufzeitschicht-Modell

- **Präsentation für einen Benutzer**
 - = **Instanziierung** einer Komponente
- **Laufzeit-Cache:**
 - Kopie der Komponente angelegt, wird angeschaut und evtl. verändert, dann zurückgeschrieben in die Speicherungsschicht
 - ggf. mehrere Instanzierungen zu einer Komponente
 - jede Instanzierung erhält eindeutige Identifikation (IID)
 - mit einer Komponente werden auch ihre Anker instanziiert:
Link-Markierung

Laufzeitschicht-Modell (2)

- **Sitzung (session):**
 - Verwaltungseinheit der Laufzeitschicht, aktuelle Zuordnung von Komponenten und ihren Instanzierungen
 - Benutzung eröffnet Sitzung auf einem Hypertext
 - Operation **Präsentieren** von Komponenten erzeugt Instanzierungen
 - können geändert werden
 - Operation **Realisierung** der Änderungen bringt sie akkumuliert in zugehörige Komponente ein
 - schließlich **Zerstörung** der Instanzierung ("unpresenting")
 - Löschen einer Komponente über eine Instanzierung entfernt auch alle anderen Instanzierungen
 - abschließend Sitzung beenden

Laufzeitschicht-Modell (3)

zu einer Sitzung gehören:

- ❑ **der Hypertext, der benutzt wird**
- ❑ **Zuordnung der IIDs der aktuellen Instanzierungen zu ihren Komponenten**
- ❑ **Geschichte**
 - Folge aller Operationen, die seit Eröffnen der Sitzung ausgeführt wurden
 - derzeit nur im Modell verwendet, um das Konzept einer reinen Lese-Sitzung zu definieren
 - sollte aber auch jeder Operationen zur Verfügung stehen, deren Wirkung durch die Vorgeschichte verändert werden könnte

Laufzeitschicht-Modell (4)

zu einer Sitzung gehören weiterhin:

- ❑ **Laufzeit-Auflöser-Funktion**
 - Laufzeit-Version der Auflöser-Funktion in der Speicherungsschicht
 - Abbildung Spezifizierer auf UIDs
 - Spezifizierer können jetzt auf Geschichte Bezug nehmen:
"die zuletzt gelesene Komponente mit Namen X"
 - muss konsistent sein mit Auflöser in Speicherungsschicht:
jeder Spezifizierer, den Speicherungsschicht auch auflösen kann, muss auf dieselbe UID abgebildet werden

Laufzeitschicht-Modell (5)

□ Instanzierer-Funktion

- erhält UID einer Komponente und Präsentationsspezifikation, erzeugt daraus Instanzierung in einer Sitzung
- muss die in der Komponente selbst gespeicherte Präsentationsspezifikation und die ihm zusätzlich übergebene kombinieren (überlagern, vereinigen, ...)
- wird aufgerufen von Operation **presentComponent** nach Auflösung eines Spezifizierers
- wird ihrerseits aufgerufen von **followLink**, und zwar für alle Komponenten, die sich aus gegebener Link-Markierung ermitteln lassen (Richtung TO oder BIDIRECT)

□ Realisierer-Funktion

- bildet Instanzierung (mit allen Änderungen) auf (neue) Komponente ab, die dann mit Operation **modifyComponent** in Speicherungsschicht eingetragen wird

Zusammenfassung

- **mächtiger als jedes existierende Hypermedia-System**
 - Mehrwege-Links, zusammengesetzte Komponenten
- **einige Konzepte als "optional" kennzeichnen**
 - Familie von zusammengehörigen Modellen, die unterschiedliche Teilmengen der optionalen Konzepte unterstützen
- **formale Spezifikation (in Z) liegt vor**
- **nützlich für die Definition von Austauschformaten für Hypertexte**
- **Beispiel (s. unten):**
 - eingeführt für Austausch zwischen HyperCard und NoteCards; direkte Umsetzung der Dexter-Konzepte in SGML-Elemente
- **geeignete Basis für die Entwicklung von Normen**

Einfaches Austauschformat

```
<hypertext>
  <component>
    <type> text </type>
    <uid> 21 </uid>
    <data> Dies ist irgendein Text ... </data>
    <anchor>
      <id> 1 </id>
      <location> d13 </location>
    </anchor>
  </component>
  <component>
    <type> text </type>
    <uid> 777 </uid>
    <data> Dies ist irgendein anderer Text ... </data>
    <anchor>
      <id> 1 </id>
      <location> 13-19 </location>
    </anchor>
  </component>
</hypertext>
```

Einfaches Austauschformat (2)

```
<component>
  <type> link </type>
  <uid> 881 </uid>
  <specifier>
    <component_uid> 21 </component_uid>
    <anchor_id> 1 </anchor_id>
    <direction> FROM </direction>
  </specifier>
  <specifier>
    <component_uid> 777 </component_uid>
    <anchor_id> 1 </anchor_id>
    <direction> TO </direction>
  </specifier>
</component>
</hypertext>
```

8.2 HyTime

- ❑ **ISO/IEC International Standard 10744:1992**
„Hypermedia/Time-based Structuring Language“ (www.hytime.org)
- ❑ **Anwendung von SGML**
 - Nutzung der Konzepte in kontrollierter Weise (bestimmte Attribute)
- ❑ **Grundgedanke:**
 - "Superklasse" für ähnliche (multimediale) DTDs, gemeinsame Eigenschaften, Vererbungshierarchie für DTDs
→ SGML **Architectural Forms**
- ❑ **modulare Struktur:**
 - Base Module: SGML
 - Location Address Module: Adressierung (Koordinaten, Attribute, Namen)
 - Hyperlink Module: Links (independent, contextual, property)
 - Finite Coordinate Space (FCS) Module: räumliche und zeitliche Anordnung
 - Event Projection Module: Projektion auf reale Dimensionen
 - Object Modification Module: Objektdarstellung

8.3 MHEG

- Multimedia and Hypertext Experts Group (www.mheg.org)**
- ❑ **MHEG-5 (Teil 5 der MHEG Standard Suite, 1998)**
 - ❑ **entscheidender Unterschied zu HyTime:**
Final Form („bits on the wire“),
also sozusagen multimediales Layout
 - für Leser, nicht für Verlage und Autoren
 - ❑ **das "PostScript" für Hypermedia**
 - wichtige Anwendung: interaktives Fernsehen
 - ❑ **Klassenstruktur**
 - ❑ **Aktuell: MHEG-8 (XML als Alternative zu ASN.1)**