Prof. Dr.-Ing. Stefan Deßloch
AG Heterogene Informationssysteme
Geb. 36, Raum 329
Tel. 0631/205 3275
dessloch@informatik.uni-kl.de

Technische Universität Kaiserslautern

Multimedia-Datenbanken

Kapitel 7: Dokumentstrukturen

Multimedia-Dokumente

- bisher einzelne Medien getrennt,
  nun zusammengesetzt: Multi-Media
- dabei wiederum:
  - Strukturen, Operationen, Editoren/Werkzeuge, Speicherung
- Dokumente:
  - sowieso schon erfasst und bearbeitet,
    jetzt mit multimedialen Anteilen
- Gliederung:
  - Präsentations- und Gestaltungs-Werkzeuge
  - Formulare
  - Dokumentablichtungen
  - zusammengesetzte Dokumente (OpenDoc, OLE)
  - verallgemeinerte Dokumentstrukturen (SGML, ODA)
  - Hypermedia (Dexter, WWW, HyTime, MHEG – Kapitel 8)
- noch ohne Datenbanksystem, dateibasiert, aber: Schemata!

Präsentations- und Gestaltungs-Werkzeuge

- Erstellung von multimedialen Präsentationen, Komposition von multimedialen Elementen (Presentation and Authoring Tools)
  - Präsentations-Software:
    - Erstellung von Folien mit multimedialen Elementen (u. a. Audio)
      - Beispiele: Microsoft PowerPoint, Asymmetrix Compel
    - zwei Komponenten: Editor, Laufzeitumgebung
  - Ikonenbasierte Gestaltungssysteme:
    - Erzeugung von graphischen Drehbüchern oder Skripten für den Kontrollfluss in Präsentationen / interaktiven Systemen mit multimedialen Elementen
      - Beispiele: Macromedia Authorware, AimTech Iconauthor
    - komplexer: Interaktionen, Verzweigungen
  - HyperCard- und Hypermedia-Werkzeuge:
    - eigenes Kapitel, s. unten
  - Zeitachsenbasierte Systeme:
    - Arrangieren von multimedialen Elementen entlang der Zeit, nacheinander oder gleichzeitig
      - Beispiele: Asymmetrix MediaBlitz, Real Presentation Maker

Präsentations- und Gestaltungs-Werkzeuge (2)

- Ergebnis
  - Datei, enthält gesamte Präsentation
- Übernahme in DB
  - als BLOB möglich
- gemeinsame Verwendung von Multimedia-Objekten in mehreren Präsentationen:
  - oft auch unterstützt (z. B. SMIL)
  - eigene Datei, Verweis über Dateinamen bzw. URL
- Einhaltung der Zeitschranken
  - nicht garantiert, System muss leistungsfähig genug sein ("best effort")
- Datenbanken
  - würden Präsentationen als komplexe Objekte modellieren – OODBS
  - fügen Metadaten und Unterstützung der Suche hinzu

(Elektronische) Formulare

- Anordnung von benannten Feldern zur effiziente Bearbeitung von Vorgängen
- Einsatzmöglichkeiten:
  - Dokumentablichtungssysteme (document-imaging systems):
    - Formulare ergänzen die „Fotos“ von Dokumenten
    - ausgefüllt von der Person, die die Dokumente einscannt, oder automatisch extrahiert
  - Datenbanksysteme:
    - Oberflächengestaltung für Anzeigen, Ändern und Suchen von Datensätzen ("query by forms") – auch mit multimedialen Elementen (siehe QBIC)
  - formularbasierte Programmpakete:
    - Zugang zu speziellen Dateien oder Erzeugung eines Formulars für ein einzelnes Dokument
    - mehr Möglichkeiten der Gestaltung als bei den Datenbank-Oberflächen, z. B. Graphik und Rasterbild
  - Groupware-Programmpakete:
    - auch noch Formular-Bearbeitung
    - z. B. in Form von E-Mail bis hin zu Workflow (Beispiel: Lotus Notes)
- Formular-Entwurf und -Erstellung:
  - Elemente: Felder, Tasten (Buttons), Graphik, Tabellen
  - Verknüpfung mit gespeicherten Datenobjekten oder Berechnungen
  - Enger Bezug zum Schema einer Datenbank: Vollständigkeit, Suchmöglichkeit

Dokumentablichtungen

(Document-imaging Systems)

- Übernahme vorhandener Papier-Dokumente
  - als Bild, Formular oder Text
  - und dazu gehöriger Vorgänge
  - in elektronische Dokumente
  - und automatische Weiterleitung
- elektronische Weiterverarbeitung
  - in verschiedensten Formen, oft mit kontrollierter Weitergabe (Workflow)
- drei Basisschritte:
  - Scannen:
    - eingehende Dokumente (Poststelle)
    - verschiedene Geschwindigkeiten (und Kosten!)
  - Indexieren:
    - je nach Typ und Einheitlichkeit der Dokumente
    - manuell oder OCR
    - ganzes Dokument oder Teile
    - anschl. Feature-Extraktion (IR-Techniken)
  - Workflow:
    - Weitergabe, Annotationen

Zusammengesetzte Dokumente

- Dateien
  - in heutigen Betriebssystemen oft getypt
  - bestimmen eindeutig die Anwendung, mit der sie gelesen und verändert werden können
- Zusammengesetzte Dokumente (Compound Documents)
  - verbinden Dateien ("Objekte"), die verschiedene Anwendungen benötigen
    - z. B. Tabelle im Text
- Interoperabilität der Anwendungen:
  - rufen sich ggf. wechselseitig auf, tauschen Daten aus
- Werden dadurch zu "Komponenten-Software":
  - wiederverwendbar, nicht nur eigenständige Anwendung, sondern auch als Unterprogramm in anderen Anwendungen (Komponente)
- Beispiele:
  - Datei-Komprimierung und -Konvertierung, Rechtschreibprüfung, Tabellenkalkulation, Textverarbeitung
- Komponenten eines Dokuments
  - können verteilt gespeichert sein – und damit auch zur Kooperation verwendet werden

Zusammengesetzte Dokumente - Normung

- Zwei konkurrierende Vorschläge für eine Norm:
  - OpenDoc von CILabs (Zusammenschluss von Microsoft-Konkurrenten)
  - OLE 2.0 von Microsoft
- Gegenstand der Normung:
  - Binärdarstellung der Objekte (Anwendungen):
    - in verschiedenen Programmiersprachen erstellt, aber einheitliche Aufrufschnittstellen
    - OpenDoc: System Object Model (SOM) von IBM auf der Basis von CORBA
    - Microsoft: Component Object Model (COM)
  - Benutzerschnittstelle:
    - Funktionen zum Erzeugen, Ausgeben und Bearbeiten zusammengesetzter Dokumente (Menüeinträge usw.)
  - Speicherung der Dokumente:
    - als Hierarchie von Bausteinen, einschl. Versionierung und Transaktionsunterstützung
  - Verteilung und Zusammenarbeit:
    - Weitergabe von Dokument-Bausteinen über Netze
    - Synchronisation der Zugriffe

OpenDoc

- unterstützt z. B. von Apple und IBM
- Quellcode
  - gehört unabhängiger Non-Profit-Organisation namens Component Integration Laboratories (CILabs)
- Dokument-Modell:
  - strukturierte Sammlung von Teilen (parts);
    also eher ein Ordner, aber mit Struktur und Schnittstellen
  - Protokolle für die Zusammenarbeit der Anwendungen, die Teile bearbeiten (bezügl. Speicherverwaltung, Ereignisweitergabe, Laufzeitsystem, Benutzerschnittstelle)
- Teile (parts):
  - können andere Teile enthalten,
    müssen aber relativ wenig über deren Inhalte wissen
  - pro Dokument stets ein Wurzel-Teil (root part),
    enthält (transitiv) sämtliche Teile des Dokuments
  - jedes Teil hat eigenes Inhalts-Modell (content model), in dem es ebenfalls "Teile" geben kann (content objects), die aber für OpenDoc keine Rolle spielen

OpenDoc (2)

- Teil-Verwalter (part handler):
  - die "Anwendungen" in der OpenDoc-Architektur
  - Anzeigen oder Bearbeiten eines Teils,
    zuständig für das Inhalts-Modell und für die Speicherung
  - Teil und Verwalter zusammen bilden "Objekt"
  - aufgeteilt in Editoren und Viewer
    - sollten zu einem Inhalts-Modell jeweils verfügbar sein
    - Editor kostenpflichtig, Viewer umsonst
- eingebettetes Bearbeiten:
  - kein Kontextwechsel
  - kein manuelles Starten von Teil-Verwaltern
- Rahmen (frames):
  - Elemente der Benutzerschnittstelle,
    Anzeige eines Teils in einem Dokument
  - reguläre Form (Rechteck), aber auch irreguläre möglich

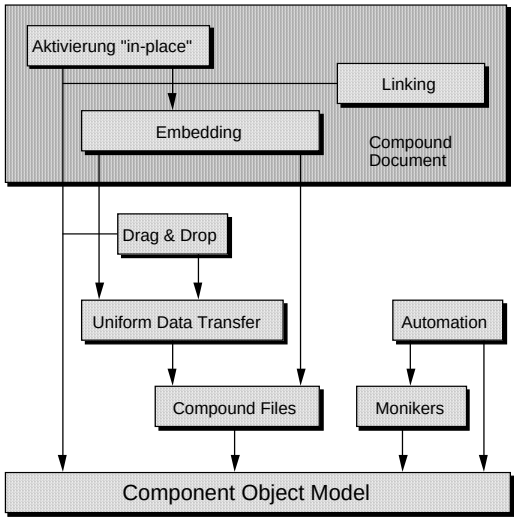
OpenDoc (3)

- Linking:
  - Teile nicht nur einbetten, sondern auch darauf verweisen (gemeinsame Benutzung in mehreren Dokumenten)
- Versionierung:
  - Benutzer können Drafts erzeugen und darin arbeiten
  - verschiedene nebeneinander, später zusammenführen
  - sichtbar auch für andere
  - Speicherung nur der geänderten Teile
- Verteilung:
  - Verwendung von IBMs Distributed System Object Model (DSOM)

OLE 2.0

- "Object Linking and Embedding"
  - verfügbar seit April 1993
  - etliche hundert Anwendungen verfügbar, die damit arbeiten
- Component Object Model (COM):
  - Interoperabilität von Objekten (Anwendungen), die in verschiedenen Sprachen entwickelt wurden
  - enthält Spezifikation und ausführbaren Code (COMPOBJ.DLL)
  - COM-Objekte bieten eine oder mehrere Schnittstellen an
  - können erfragt werden – Objekt liefert dann Handle für alle Operationen, die es ausführen kann
- auch verteilt:
  - Schnittstellen von Objekten auf anderen Rechnern aufrufen
- Speicherungsmodell:
  - Speicherobjekt (= Datei) intern Mehrweg-Baum: Stream = Blattknoten, Substorage = innere Knoten ("Dateisystem in einer Datei")

OLE-Architektur



OLE-Konzepte

- Einbettung von Objekten:
  - als Stream in derselben Datei
- Ansehen von Komponenten:
  - auch möglich, wenn Anwendung lokal nicht verfügbar: automatische Erzeugung einer Graphik des Objekts
  - Änderung verlangt allerdings dann doch die Anwendung (Wurzel-Objekt auch)
- Linking und Monikers:
  - verknüpfte Objekte anderswo gespeichert (z. B. in einem anderen zusammengesetzten Dokument)
  - statt absoluter Dateinamen (wie in OLE 1.0) nun Monikers: Ort plus Code zur Bindung – stabiler
- Verteilung:
  - unsichtbar für die Anwendung durch Installation und Verwendung neuer DLLs
  - inkl. Deadlock-Behandlung

Verallgemeinerte Dokumentstrukturen

- bisher: abstrakte Strukturen und technische Lösungen stark vermischt
- MMDBVS: Strukturen wichtig, auf Schema übertragen
- Verallgemeinerte Dokumentstrukturen
  - systemübergreifend!
  - Meta-Ebene
    - Datenmodell für Dokumente
    - Definition von Dokumenttypen
  - hier:
    - SGML
    - XML
    - ODA

SGML

- "Standard Generalized Markup Language"
  - Beschreibung der Struktur von Dokumenten (u. a.)
  - Sprache zur Definition einer Syntax (Meta-Sprache)
  - Datenbeschreibung für den Datenaustausch
- Entwicklung:
  - GML ab Ende 60er Jahre bei der IBM unter der Leitung von Charles Goldfarb
  - daraus im Auftrag des American National Standards Institute (ANSI – heute NIST, National Institute for Standards) unter führender Teilnahme Goldfarbs SGML entwickelt
  - 1986 als ISO-Norm 8879 veröffentlicht
- kein Layout – nur inhaltliche Komponenten
  - sinnvoll, wenn Layout unbedeutend oder Dokument für mehrere, verschiedene Layouts benötigt wird
  - Beispiele: Archivierung, Austausch zwischen Verlagen, technische Dokumentationen (z. B. DocBook)
- Beschreibung der logischen Struktur mittels Auszeichnungen (markups)
  - nicht vorgegeben, sondern selbst definierbar
    - deshalb "verallgemeinert" – generalized

SGML (2)

- Dokumenttypen = Sätze von Auszeichnungen mit Regeln
  - aus welchen Elementen müssen Texte eines Typs bestehen?
  - welche müssen sein und welche können entfallen?
  - wie verschachtelt und welche Reihenfolge?
- dann Werkzeuge:
  - Parser zur Prüfung der syntaktischen Korrektheit eines Dokuments
  - Konverter
    - in andere SGML-Dokumenttypen, Druckaufbereitung mit TeX o. ä.

SGML (3)

Beispiel: Gedicht

- Elemente:

```
<!ELEMENT zeile - - (#PCDATA)>
<!ELEMENT strophe - - (zeile+)>
<!ELEMENT gedicht - - (strophe+)>
```

- Textersetzungen:

```
<!ENTITY auml SDATA "[auml]">
<!ENTITY ouml SDATA "[ouml]">
    ....
<!ENTITY szlig SDATA "[szlig]">
```

(für "[auml]" kann auch "ä" oder """a" stehen)

- Attribute:

```
<!ELEMENT reim - - (zeile+)>
<!ELEMENT kehr - O EMPTY>
<!ATTLIST reim nummer ID #REQUIRED>
<!ATTLIST kehr nummer IDREF #REQUIRED>
```

```
<reim nummer=reim23>
    <zeile>Fidirallala, fidirallala,
      fidirallarallala</zeile>
</reim>
    ....
<kehr nummer=reim23>
```

SGML (4)

- beliebige (hierarchische) Strukturen beschreibbar
- DTD – Document Type Definition:
  - entscheidend für gemeinsames Verständnis
- verbreitete Anwendung:
  - HTML
- Erweiterung für Multimedia:
  - HyTime
- zunehmend abgelöst durch XML
  - vereinfachte Version von SGML
  - XHTML: Redesign von HTML
  - SMIL: Synchronized Multimedia Integration Language

ODA

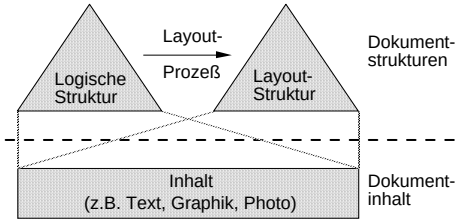
"Open Document Architecture"

- ebenfalls für den Austausch von Dokumenten zwischen beliebigen Systemen
- "Architektur": allgemeiner Aufbau von Dokumenten
- davon ableitbar: Familie von Dokumentaustauschformaten (Open Document Interchange Format – ODIF)
- Sept. 1985: ECMA Standard 101,
  April 1986: ISO Draft International Standard 8613

- Dokumentarchitektur-Modell
  - jede gewünschte Dokumentbearbeitungsfunktion soll in der Modellwelt beschreibbar sein
  - leichte Bearbeitbarkeit gewährleisten
  - Umsetzung zwischen Formaten nach dem Modell ohne Informationsverlust möglich

ODA (2)

- Logische Struktur:
  - Kapitel, Abschnitte, Sätze, Bilder
- Layout-Struktur:
  - Seiten, Satzblöcke (Rechtecke)
- Inhalt:
  - Text, Bild usw. – "Bausteine"
- Trennung von Struktur und Inhalt
  - Strukturen nicht durch eingebettete Steuerzeichen definiert, sondern explizit durch Hierarchie von Objekten
  - jedes logische oder Layout-Objekt: Instanz einer Objektklasse



ODA (3)

- Dokument selbst: Dokumentklasse
  - Definition von Objektklassen ("Regeln")
  - vorgegebene Inhaltsstücke (generic content), z. B. Firmenlogo, Standard-Paragraphen
- Objekttypen
  - in der Norm festgelegt
  - bestimmte Attribute auf sie anwendbar
- logische Struktur:
  - Document Logical Root: Wurzel des Baums
  - Basic Logical Object: Blätter mit "Inhaltsportionen"
  - Composite Logical Object: Zwischenknoten
- Layout-Struktur:
  - Document Layout Root: Wurzel des Baums
  - Page Set: Gruppe von Seiten
  - Page: zweidimensionaler Bereich, Seite
  - Frame: rechteckiger Bereich auf einer Seite
  - Block: formatierter Inhalt nur eines Mediums

ODA (4)

- Objektklassen
  - in der Dokumentklassendefinition
    auf der Basis von Objekttypen definiert
  - z. B. Paragraph, Fußnote, Bildunterschrift
    vom Typ "Basic Logical Object"
    Header Frame, Column Frame, Footer Frame
    vom Typ "Frame"
- Inhaltsstücke (content portions)
  - Teile des Dokuments
  - durch logische und Layout-Struktur definiert
- Content Architecture
  - für jede Art von Inhalt (Text, Graphik, Rasterbild) eine
    maschinen-unabhängige Codierung
- ausgetauschte Nachricht enthält
  - neben logischer Struktur, Layout-Struktur und Inhaltsportionen
  - auch die Klassenbeschreibung (Generic Logical Structure, Generic Layout Structure)
  - d. h. Schemainformation mitgeführt, "selbstbeschreibend"