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

Technische Universität Kaiserslautern

Chapter 7 - Multimedia Documents

Outline

- Special document structures
  - presentation and authoring tools
  - electronic forms
  - compound documents
- Generalized document structures
  - SGML (XML)
  - ODA
- Hypertext and hypermedia

Presentation and Authoring Tools

- Used to create MM presentations, composition of MM elements
  - presentation software (editor & viewer/presenter)
    - presentation charts that include MM elements (audio, image, video)
      - examples: Microsoft PowerPoint, Asymmetrix Compel
  - icon-based authoring tools
    - creation of graphical screenplays/scripts for describing complex control flow and interactions in presentations, e-learning applications, etc.
      - examples: Macromedia Authorware, AimTech Iconauthor
  - timeline-based systems
    - arranging MM elements along presentation time axis, sequential or parallel presentation
      - examples: Asymmetrix MediaBlitz, Real Presentation Maker
  - hypercard, hypermedia systems (see discussion later in this chapter)
- Result
  - file that contains the complete presentation
  - MM elements may be shared across multiple presentations (e.g., SMIL, discussed later)
    - element stored in separate file, referenced by multiple presentations

(Electronic) Forms

- Consist of named fields (in a layout) for efficiently processing specific tasks
- Application areas
  - document imaging systems
    - form enhances the document "image"
    - field entries provided manually or extracted from document
  - database systems
    - user interface for search, result presentation, editing of records ("query by forms") – may include MM elements (e.g., QBIC)
  - groupware systems
    - includes form editing/authoring
    - e.g., based on email, workflow support (example: Lotus Notes)
- Design and Creation
  - elements: fields, buttons, graphics, tables
  - linkage to stored data objects, computations
  - similarity to database schema: completeness, uniformity, search capabilities

Compound Documents

- Files
  - usually typed in modern OS
  - associated with application for viewing, printing, editing
- Compound Documents
  - composition of files ("objects") that require different applications
    - e.g., table/spreadsheet within text
- Interoperability of applications
  - invocation of functionality, exchange of data
- Support of "component software"
  - not just standalone, but also reusable as a component (subroutine) in another program
- Examples: file compression, file conversion, spell-checking, spreadsheet capabilities
- Document components may be stored in a distributed manner (-> cooperation)
- Standards
  - OLE 2.0 by Microsoft
  - OpenDoc by CILabs (consortium of Microsoft competitors)

Generic Document Structures

- So far: no separation of abstract structures and technical solutions
  - "black box" for content repository, DBMS
- Generic document models and structures
  - independent of systems, platforms
  - meta-level
    - data model for documents
    - definition of document types
  - here:
    - SGML
    - XML
    - ODA

SGML

- "Standard Generalized Markup Language"
  - Description of document structure
  - language for defining a syntax (meta-language)
  - data description for data exchange
- History
  - GML originated in the late 60's at IBM (Charles Goldfarb)
  - used as a basis for developing SGML
    - American National Standards Institute (ANSI – today NIST, National Institute for Standards)
    - led by Charles Goldfarb
  - published as ISO-Standard 8879 in 1986
- No layout – only content/structure
  - useful if layout is not important, or document needs to be published with multiple layouts
  - examples: archiving, document exchange, technical manuals (e.g., DocBook)
- Markups for describing the logical document structure
  - not predefined, can be defined separately
    - "generalized" markup language

SGML (2)

- Document type: vocabulary and rules for specific markup
  - document type definition – DTD
  - what elements does the document consist of?
  - mandatory vs. optional elements?
  - possible order and hierarchical structure (nesting)
  - example: HTML, HyTime (for multimedia)
- Tools
  - parser for checking syntactic correctness (validity) of documents
  - converter
    - into other SGML document type, TeX, ...
- XML has "taken over"
  - simplified version of SGML
  - XHTML: redesign of HTML
  - SMIL: Synchronized Multimedia Integration Language

Synchronized Multimedia Integration Language

- Creations of multi-media presentations, declarative description of
  - presentation layout
  - objects involved
  - timing aspects
- SMIL is **not** a container format
  - multi-media objects (pictures, audio, video, …) are not included, only referenced via a URI
    - same object can be included in multiple presentations
    - integration of "remote" objects (e.g., weather chart)
- Object alternatives in SMIL
  - same object in different resolutions, dynamically selected based on available bandwidth
  - text or audio stored in different languages, selected based on user language preferences
- Interactive capabilities (limited)
  - follow presentation links

SMIL Documents

- XML documents following the SMIL DTD or Schema
  - document header
    - general presentation properties, such as layout, position of regions, …
    - based on a layout language close to cascading style sheets (CSS)
  - document body
    - actual definition of the presentation (incl. timing aspects)
    - number of XML element tags for different types of MM-objects
      - <audio>, <img>, <video>, <textstream>
      - Attributes for representing object URI, presentation region, MIME-type, presentation duration
    - objects can be "hyperlinked" to other resources on the web
    - timing aspects
      - parallel presentation (i.e., at the same time)
      - sequential presentation
      - duration attribute

Example:

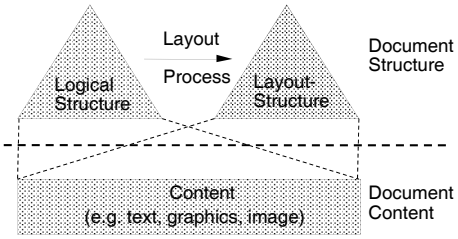
```
<seq>
    <img id="a" dur="6s" begin="0s" src="..." />
    <img id="b" dur="4s" begin="0s" src="..." />
    <img id="c" dur="5s" begin="2s" src="..." />
</seq>
```

ODA

- "Open Document Architecture" (ODA)
  - goal: enable/facilitate exchange of documents among arbitrary programs/systems
  - "architecture": generic model for document structure
  - derived from generic architecture: family of document exchange formats (Open Document Interchange Format – ODIF)
  - Sept. 1985: ECMA Standard 101,
    April 1986: ISO Draft International Standard 8613
  - not successful as a standard, but strong influence on other efforts
    - HTML/CSS, XML/XSL, OpenDocument (OASIS Open Document Format for Office Applications)
- Document architecture model – goals
  - capture arbitrary operations for document manipulation
  - enable ease of manipulation
  - enable document format transformation without losing information

ODA (2)

- Logical structure
  - chapter, section, paragraph, figure, ...
- Layout structure
  - page, block (rectangle), ...
- Content portions
  - text, image, graphics etc.
- Separation of structure and content
  - structure is made explicit using a hierarchy of objects (i.e., not using embedded special characters)
  - every logical or layout object is an instance of an object class



ODA (3)

- Document itself: document class
  - definition of object classes ("rules")
  - generic content, e.g., company logo, standard paragraph
- Object types
  - defined by the standard
  - with a set of applicable attributes
- Logical structure
  - Document Logical Root: root of logical object tree
  - Basic Logical Object: leaves with "content portions"
  - Composite Logical Object: intermediate nodes
- Layout structure
  - Document Layout Root: root of layout object tree
  - Page Set: group of pages
  - Page: 2D area
  - Frame: rectangular area within page
  - Block: formatted content of a single media type

ODA (4)

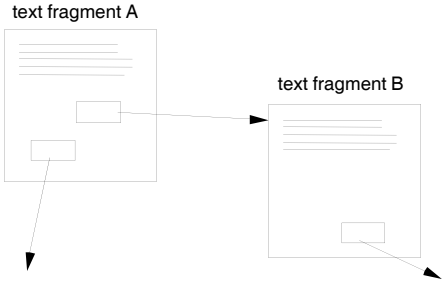
- Object classes
  - defined in document class definition based on object types
  - e.g., paragraph, footnote, figure title of type "basic logical object"
    header frame, column frame, footer frame of type "frame"
- Content portions
  - components of the document
  - defined through logical and layout structure
    - content portion belongs to one logical and one layout object (if layout is defined)
- Content architecture
  - machine-independent format for every type of content (text, graphics, image)
- Document exchange includes
  - logical structure, layout structure, content portions
  - class description (generic logical structure, generic layout structure)
  - i.e., schema information is included, document is "self-describing"

Hypertext

- Departure from (sequential) paper form
- Primary form of "mechanical" document organization (libraries, archives)
- Starting point: Memex-System [Bush45a] based on micro-film
- Computer-based (early 60s)
  - D. Engelbart: NLS/Augment
  - T. Nelson: Xanadu project
- Became popular through
  - Apple's HyperCard on Macintosh (1987)
  - and of course the World-Wide Web (1991)
- What is hypertext?
  - collection of text fragments (articles, notes, ...)
  - system-supported connections among them (reference, cross-reference, link)
  - users are following the connections/links and keep building new ones: "non-linear text", "information web"

Hypertext (2)

- User interface:
  - windows on the screen (one for each node), showing fragments of text
  - link or anchor is distinguished visually (icon, bold, color, underlined, ...)
  - mouse-click on link opens new window with the referenced text fragment
- Data organized as a network of nodes (text fragments)
  - nodes have unique names
  - arbitrary number of "link icons" per window
  - different link types: link icon labels
  - flexible creation of new nodes and links (annotations, comments)

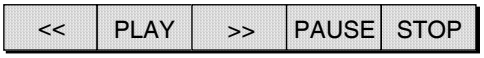


Hypertext (3)

- Search
  - follow the links (browse, navigate)
  - search hypertext network based on text (keywords, attribute values)
  - graphical representation of the network, with clickable node "miniatures"
- Importance of a browser
  - network of text fragments may become very large ("lost in hyperspace")
  - graphical network representation
  - keeping track of search/navigation path ("where was I?")
  - today, "browser" has a different/more restrictive meaning
    - web client, less functionality
- Hypertext can be seen as
  - method for data organization (similar to data model), even for storage (links as references or pointers)
  - presentation method and information model (similar to semantic networks, ER), requiring a mapping to a data/storage model
  - user interaction model, operating on underlying data/storage structures

Hypermedia

- Extension of the hypertext model
- Nodes
  - may contain graphics, images, sound, videos
- "Dynamic" media (time dependency)
  - indirect representation using "placeholder" nodes
    - showing symbol or icon for media object (speaker, display, ...)
  - short summary as text
    - remarks by xyz regarding topic ...", "film about the history of ..."
  - information about duration!
  - key pad:



Hypermedia – Important Systems

- Intermedia
  - Brown University (Providence, Rhode Island)
  - support for teaching and research in a university setting
- NoteCards
  - Xerox PARC
  - "index cards" for (technical) reports
- Neptune / HAM
  - Tektronix
  - frontend – backend, transaction concept
- Hyperties
  - University of Maryland
  - IBM PC
  - education and teaching, kiosks, museums
- KMS
  - Knowledge Systems
  - successor of ZOG (Carnegie-Mellon University)
  - no windows! 1-2 nodes per screen
- HyperCard
  - Apple
  - direct support for graphics/images on the cards
  - audio recording as a "resource", i.e., separate file
  - HyperTalk command "play *filename*"

Dexter Reference Model

- Purpose of the Dexter reference model:
  - system comparison
  - exchange, interoperability
- Three layers:
  - **storage layer:** network of nodes and links
  - **within-component layer:** content and structure of nodes
  - **run-time layer:** interaction of users with the system
- Storage layer: core of the reference model
  - includes mechanisms for the specification of anchors and presentation, which represent the interfaces to the other layers
  - data base, consisting of (atomic) **components**, connected via **links**
  - components correspond to nodes
    - deliberate choice of a different, system-neutral terminology
    - generic data containers, not concerned with inner structure

Simple Storage Layer Model

- Hypertext =
  - finite set of components
  - with two functions: **resolver** and **accessor** for finding components
    - mapping of a component specification to the component itself
- Components:
  - **atomic** components:
    - correspond to nodes in most systems
  - **links**:
    - relationships/connections between other components
    - sequence of one or mode "endpoint specifications", referring to parts of components
  - **composite** components:
    - contain other components
    - structure is a directed, acyclic graph, i.e., there may be shared components but no cycles

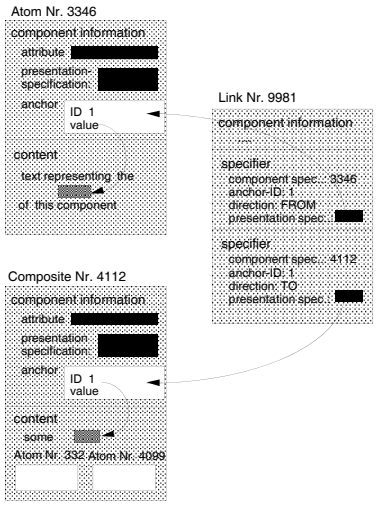
Storage Layer Model (2)

- Globally unique identification: unique identifier (UID)
  - beyond single hyper documents
  - **accessor** has to support returning the associated component for every UID
- UIDs are foundation for addressing, but UIDs alone are not sufficient
  - references to other components should also be supported based on their properties, e.g., text containing a specific word – may result in zero or multiple components
  - requires **component specifications** within the links, which need to be handled by the **resolver** function
    - UID is permitted, too – resolver is equivalent to identity in that case
- Links between parts of components:
  - UID alone is not sufficient → **anchor** consisting of ID and value
  - ID is unique within the component
  - value specifies location, region, entry, etc. within a component in an arbitrary manner, interpreted by the application, may change

Storage Layer Model (3)

- Specifier
  - Anchor-ID is associated with a component specification
  - contains
    - direction: FROM, TO, BIDIRECT, NONE
      - NONE is used, if the anchor is actually not a part of the component, but a program or script
    - presentation specification (see discussion below)
- Link
  - sequence of 2 or more specifiers
  - very generic: **n-way links**
  - only restriction: at least one specifier with direction TO or BIDIRECT has to exist
- Component
  - represented by its content
  - and **component information**:
    - sequence of anchors
    - presentation specification
    - set of attribute-value-pairs
  - attributes are arbitrary, e.g., keyword(s) or type

Storage Layer Model (4)



Storage Layer Model (5)

- Operations:
  - Add, delete a component
  - Modify anchor or attribute value
  - Retrieve component based on UID or specifier
  - **LinksTo**: maps UID of a component to the UIDs of all links that specify this component
  - **LinksToAnchor**: maps an anchor to the UIDs of all links that specify this anchor
- Integrity constraints (invariants):
  - accessor function (UID → component) has to have an inverse function, i.e., all components have a UID
  - resolver function needs to be able to potentially return all valid UIDs
  - no cycles in the composition structure, i.e., no component contains itself
  - link consistency: component specification has to reference an existing component (i.e., referential integrity)

Within-Component Layer

- no details defined in the Dexter model, could be anything
- other reference models could be used, combined with the Dexter model
  - e.g., ODA, IGES (www.nist.gov/iges/) usw.
- requires interface between hypermedia network and component content: addressing of locations and elements *inside* the component – **anchoring**
  - links between parts of nodes, both for source and target of the link

Run-Time Layer

- Hypermedia systems do not only provide passive structures, but also tools for access, viewing, modification
- Dexter model supports this aspect only in a very basic manner
- Interface required: **presentation specification**
- Include information about the presentation of components and networks in the storage layer
- Presentation is not only determined by the presentation tool, but also by the component itself and/or the link that led to it
  - example: reference to an animation in a teaching environment
    - start a *viewer* for regular users
    - start an *editor* for author/teacher

Simple Runtime Layer Model

- Presentation for an end user
  - corresponds to a component **instantiation**
- Run-time cache
  - manages a copy of the component for viewing, modifications, to be later written back to the storage layer
  - there may be multiple instantiations of the same component
  - every instantiation has a unique ID (IID)
  - together with the component, its anchors are instantiated: **link marker** – visual representation of an anchor in the presentation
- Session
  - management unit of runtime layer, keeps association of components and instantiations
  - user starts a session on a hypertext
  - operation **present component** performs instantiations, which may be modified
  - operation **realize modifications** updates the corresponding component
  - finally, **unpresent instantiation** destroys the instantiation
  - deleting a component through its instantiation will remove all existing instantiations
  - user ends/closes the session

Runtime Layer Model (2)

Information related to a session:

- hypertext used in the session
- relationship of IIDs of current instantiations to their components
- **history**
  - sequence of operations carried out since the start of the session
  - currently used in the Dexter model to define the concept of a read-only session
  - should be available to any operation whose effects may depend on the session history
- **runtime resolver function**
  - runtime version of the resolver function of the storage layer
  - maps specifiers to UIDs
  - specifier can now refer to the history: "last accessed component with name X"
  - has to be consistent with the resolver of the storage layer: a specifier that can be resolved by the storage layer has to be resolved to the same UID by the runtime layer

Runtime Layer Model (3)

- **Instantiator function**
  - receives UID of a component and a presentation specification, creates a corresponding instantiation in a session
  - needs to combine the input presentation specification with the one contained in the component (select, combine, ...)
  - called through operation **presentComponent** after specifier is resolved
  - is in turn called by **followLink** for all components that can be determined based on the link marker (direction TO or BIDIRECT)
- **Realizer function**
  - returns a (new) component that reflects the current state of the instantiation (including all edits), can be handed to the storage layer using the **modifyComponent** operation

Dexter Reference Model - Summary

- more powerful than any existing hypermedia system
  - multi-way links, composite components
- some concepts declared/regarded as "optional"
  - family of related models, supporting different subsets of optional concepts
- includes a formal specification (in Z)
- useful for the definition of hypermedia exchange formats
- example
  - exchange between HyperCard and NoteCards; directly represent Dexter concepts as SGML elements
- useful foundation for standards