

# Object-oriented hypermedia system design – a Dexter-based approach

---

*Kaj Grønbæk*

Computer Science Department  
Aarhus University  
Ny Munkegade, Bldg. 540  
DK-8000 Århus C, Denmark  
e-mail:  
kgronbak@daimi.aau.dk

*Randall H. Trigg*

Xerox Palo Alto Research Center  
3333 Coyote Hill Rd.  
Palo Alto, CA 94304, USA  
e-mail: trigg@parc.xerox.com

# Goals of tutorial

---

Leaving this tutorial, you'll hopefully have gained a sense for how the Dexter model and experiences can be applied to:

- the design of object oriented architectures for hypermedia systems;
- the enhancement and integration of existing hypermedia systems;
- the augmentation of existing application environments with linking support.

# Plan

---

1. Introduction
2. The original Dexter model
3. Improving the Dexter model
4. Object oriented design based on Dexter
5. Architecture issues
6. Special topics:
  - Cooperative hypermedia
  - Extensibility and tailorability
  - Third party application integration (via AppleEvents, DDE, etc.)
7. Wrap up

---

# Introduction

# The Dexter meetings

---

**Organizers: John Leggett and Jan Walker**

**Participants included: Rob Akscyn, Doug Engelbart, Steve Feiner, Mark Frisse, Frank Halasz, Don McCracken, Norm Meyrowitz, Tim Oren, Amy Pearl, Catherine Plaisant, Mayer Schwartz, Karen Smith, Randall Trigg, Bill Weiland.**

**Systems represented included: Augment, Concordia/Document Examiner, IGD, FRESS, Intermedia, HyperCard, Hyperties, KMS/ZOG, Neptune/HAM, NoteCards, the Sun Link Service, Textnet.**

**The meetings included:**

- **October 1988, Dexter Inn, Sunapee, New Hampshire**
- **March 1989, “Chain-o-lakes,” southeastern Texas**
- **April 1990, Cannon Beach, Oregon**
- **July 1990, Zen Retreat Center, Green Gulch, California**

# Dexter issues

---

Longstanding hypermedia issues taken up by Dexter:

- unifying the notions of node and link
- augmenting link-based networks with other structures
- integrating hypermedia with existing environments and content editors
- representing and enforcing different data models and run-time behavior
- storing properties of node presentations
- interchanging hypertexts across systems

And even defining what *counts* as hypertext / hypermedia.

# Motivation

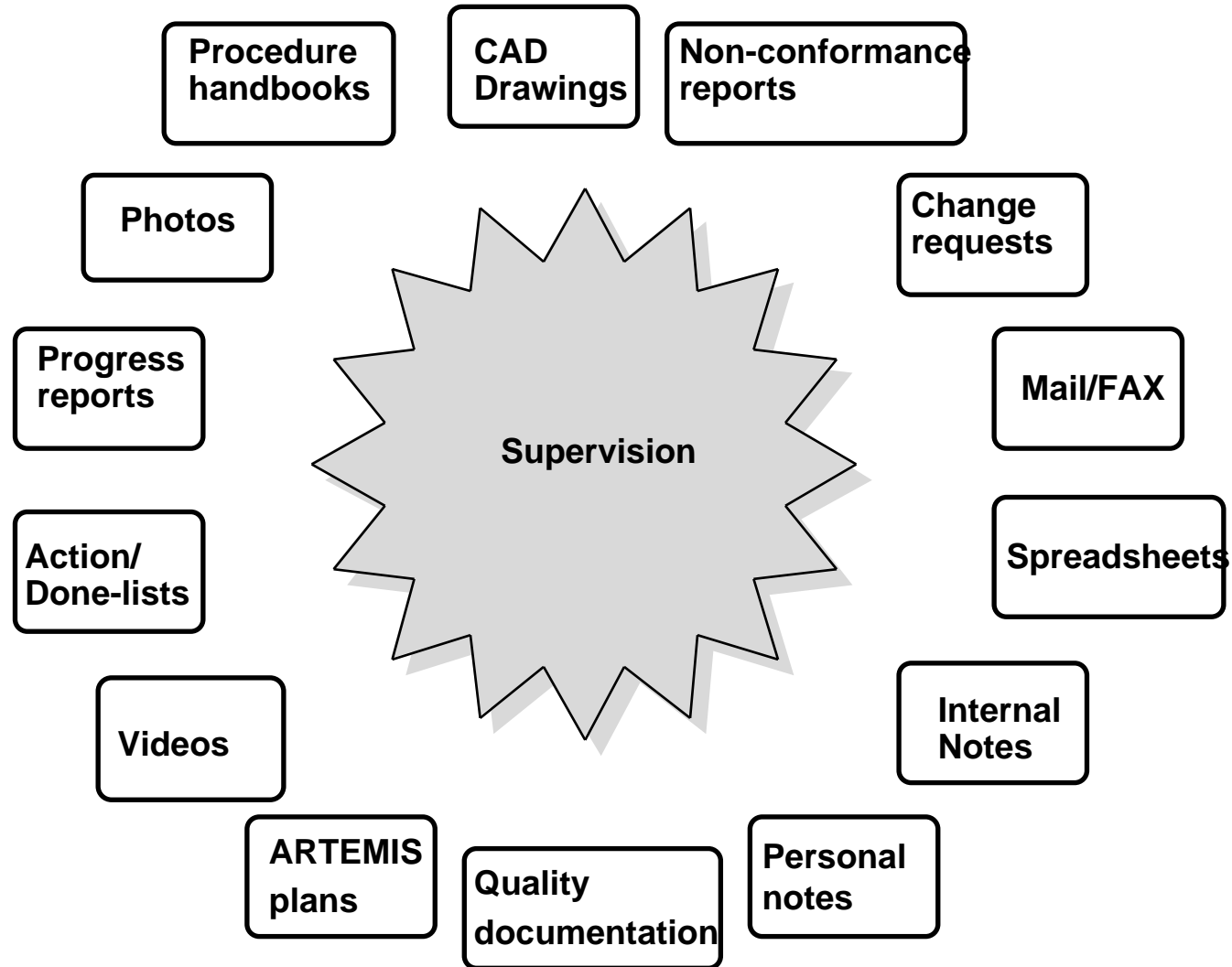
---

**Our goals for “industrial-strength” hypermedia:**

- **develop distributed, tailorable, extensible hypermedia architectures supporting cooperative work including authoring, case handling etc.;**
- **support integrated management of heterogeneous materials including text, graphics, video, design diagrams, and programs;**
- **support hypermedia integration of third party applications;**
- **support day-to-day work practices at large-scale real world work sites**

# Materials in use by supervisors at Great Belt Ltd.

---





# Distributed case handling at Great Belt Ltd.

---

Supervisors *on site* work with

- Notes
- Annotation of CAD drawings
- Taking photos and video of bridge elements
- ...

Consultants in *main office* work with

- CAD drawing database
- Work procedure handbooks
- Quality document database
- Lab tests
- ...

Supervisors *in site office* work with

- Plans and Reporting
- Action lists
- CAD drawings
- Quality document database
- ...

***GB needs support for cooperation on shared materials being worked on at different locations***

---

# Original Dexter model

# Dexter Layers - conceptual (Halasz, 1990)

---

File Name : layers75.eps

Title : /tmp\_mnt/tilde/trigg/Dexter/ECHT-tutorial/nist-slides/layers.ps

Creator : trigg

CreationDate : Wed Aug 31 01:04:31 1994

Pages : 1

# Dexter Layers - pictorial (Halasz, 1990)

---

File Name : layers2X.eps

Title : /tmp\_mnt/tilde/trigg/Dexter/ECHT-tutorial/nist-slides/layers2.ps

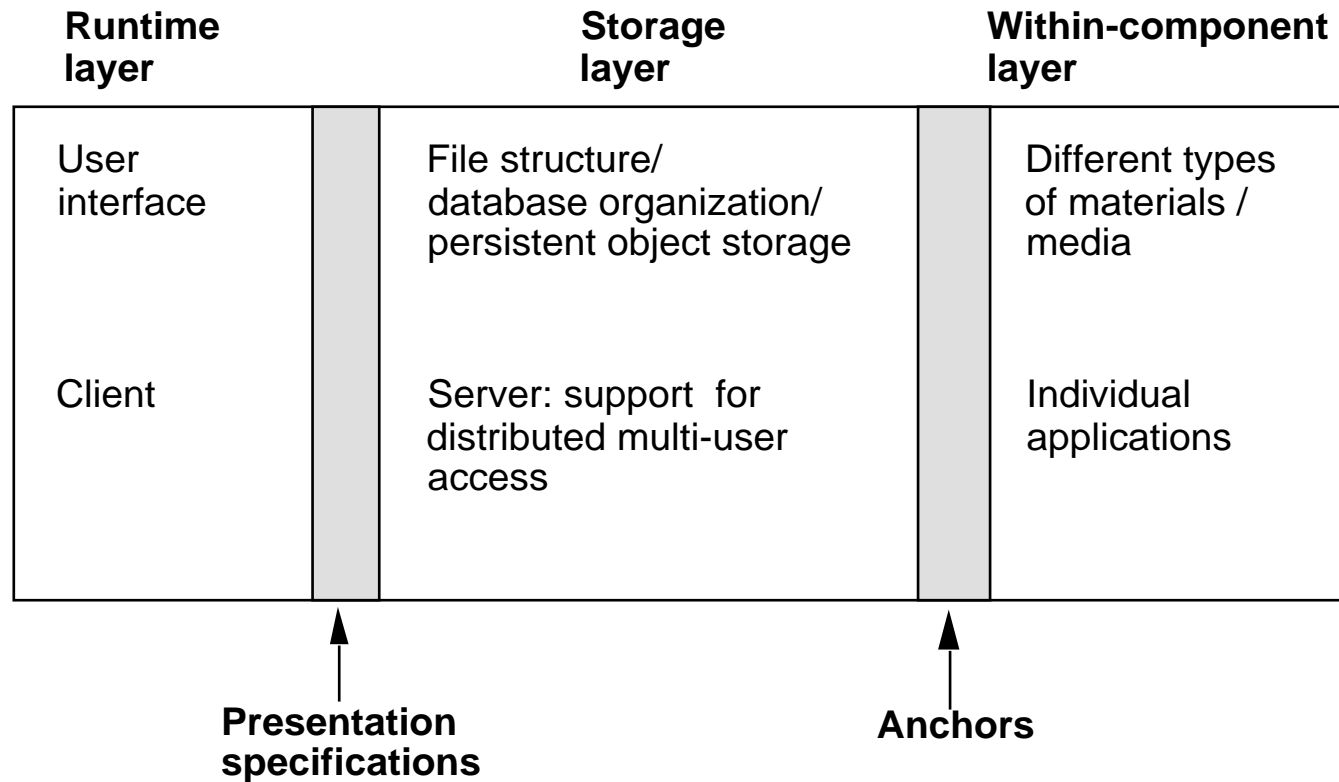
Creator : trigg

CreationDate : Wed Aug 31 01:06:16 1994

Pages : 1

# The Dexter Hypertext Reference Model

---



# Overview of Dexter terminology

Runtime Layer	Storage Layer	Within-Component Layer
Session	Hypertext	
Instantiation	Component Base Component	?
LinkMarker	Atomic Component	
	Link Component Specifier	
	Composite Component	
	Anchor	
	Presentation Specification	

# Supporting link endpoints inside document contents

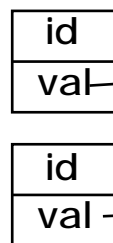
- Many hypermedia systems use embedded addresses and go-to.  
NLS/Augment, KMS, HyperCard, World Wide Web, ...

WWW/  
HTML:

```
<H2>Conference pointers </H2>
<UL>
<LI>Info about <A HREF = "http://cpsr.org/cpsr/conferences/pdc94">PDC
'94.</A>
....
</UL>
```

- Some hypermedia systems use 'anchors' stored apart from the content.
  - Intermedia, DHM, Microcosm, Multicard, PROXHY, MacWeb, ABC, WEBSs, ...

Anchor  
objects:



Advance Program:  
Participatory Design Conference (PDC'94)  
October 27-28, Chapel Hill, NC, USA  
PDC'94 is sponsored by Computer  
Professionals for Social Responsibility  
(CPSR) and in cooperation with ACM  
SIGCHI.

# 'Anchors' in recent hypermedia work

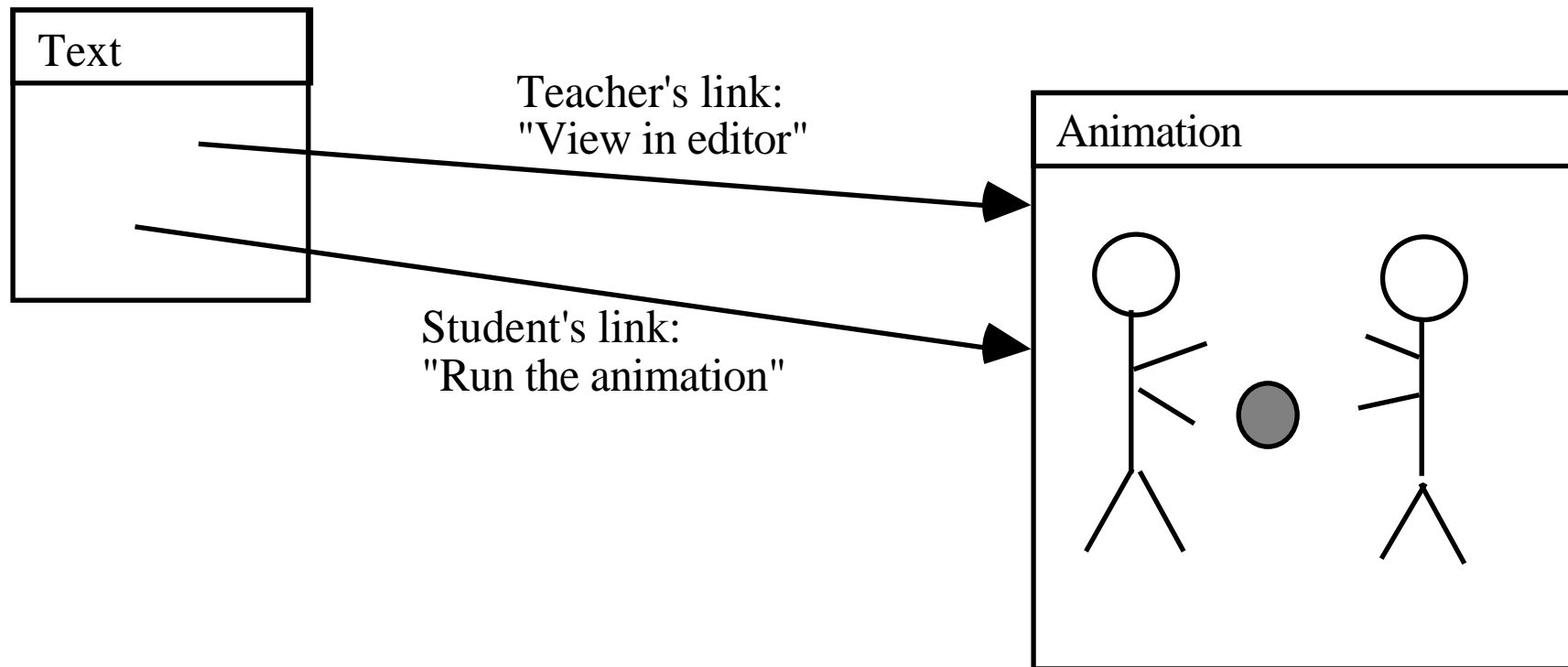
---

- Microcosm (Davis et al, ECHT'92): "In Microcosm, link source and destination anchors are recorded in terms of absolute position within a document." Indeed, they mention several of the "dangling" problems we've discussed here.
- Multicard (Rizk & Sauter, ECHT'92): "An anchor *represents* a sensitive portion of the content of a node. The associated anchor is the hypermedia object that carries the links, scripts, and other hypermedia properties. The sensitive portion is editor dependent.
- Proxhy (Kacmar & Leggett, TOIS, Oct 1991): "... *anchors* connect *application objects* to *links*. *Links* are connectors among anchors. Together links and anchors are used to represent associations among the application objects in a hypertext." Though anchors in PROXHY are actually processes, they still qualify as being represented externally from the nodes.
- MacWeb (Nanard & Nanard, HT'93): "...an **Anchor** is a reference to a small part of a text that denotes a valid concept."
- ABC (Shackelford, Smith & Smith, HT'93): "An anchor identifies part of a node's content, such as a function declaration in a program module, a definition in a glossary, or an element of a line drawing. An anchor can be used to focus an HS-link onto a specific place within the content of a node."
- WEBSs (Monnard & Pasquier-Boltuck, ECHT'92) where the term "block" is used: "A block represents any selection made inside a document (e.g. a string of characters in a textual document). The connection between two blocks is created by a link, which the user may follow..."
- Chimera (Anderson, Taylor, Whitehead, ECHT'94): "An anchor tags some portion of a view as an item of interest. Anchors are tailored by a viewer to the particular view of the object being displayed."
- HyTime (DeRose & Durand, *Making Hypermedia Work*, 1994, p. 110): "Hyperlinks connect data items known as anchors. ...an anchor can literally be anything that can be unambiguously located by a human or computer using a formal or informal notation."



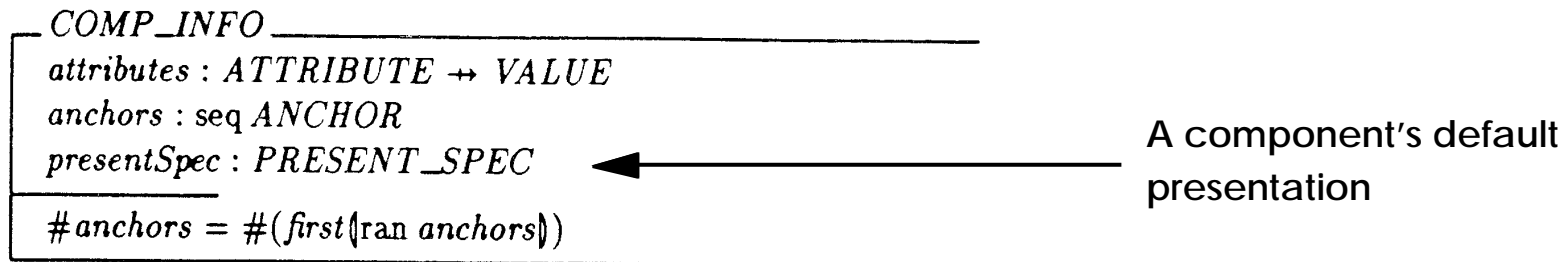
# Use of Presentation Specifications

---

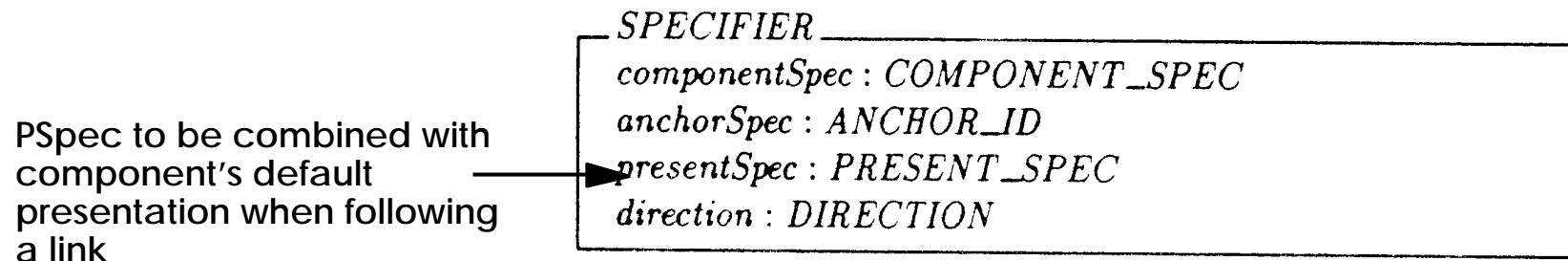


# Presentation Specifications

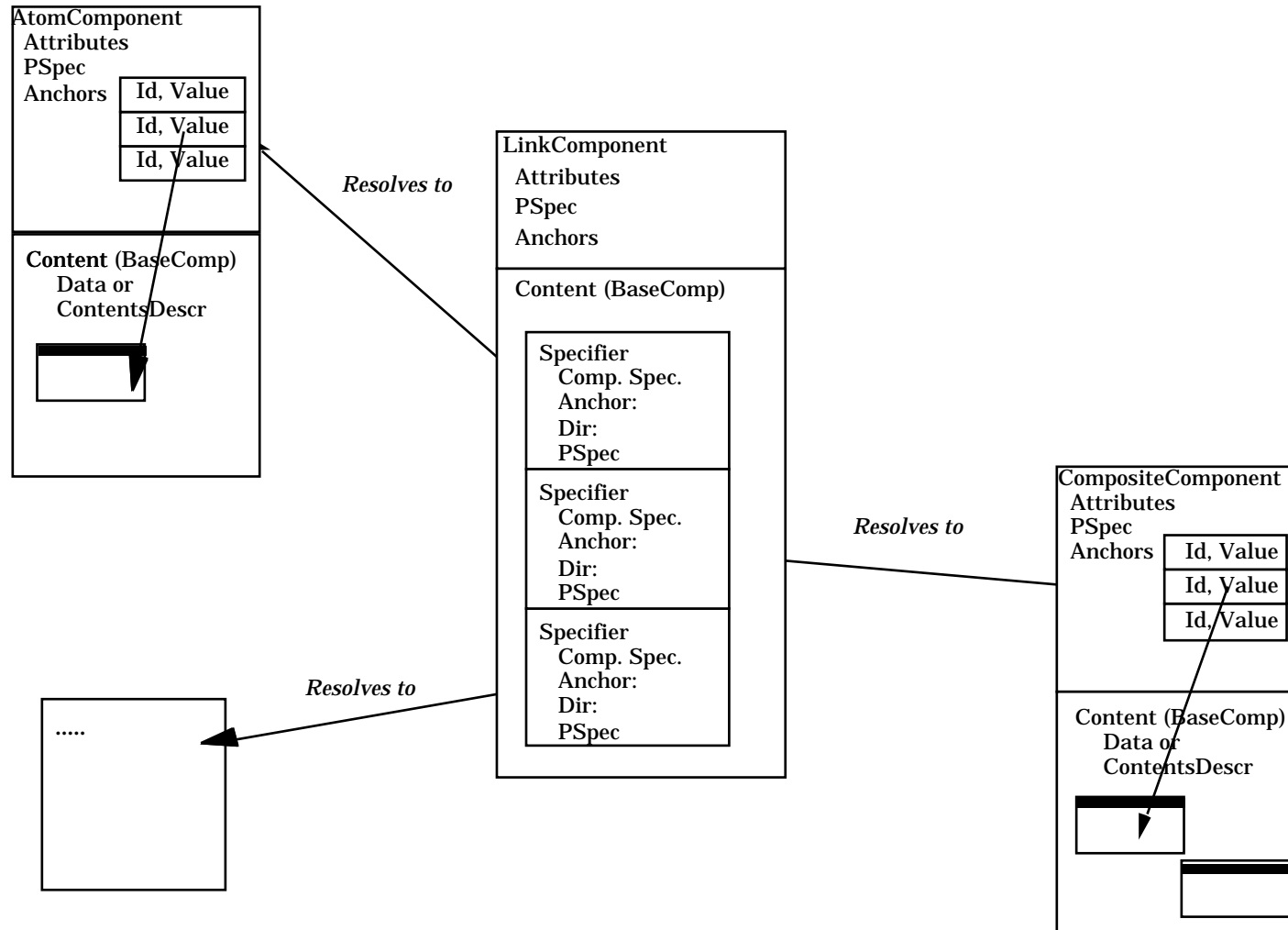
---



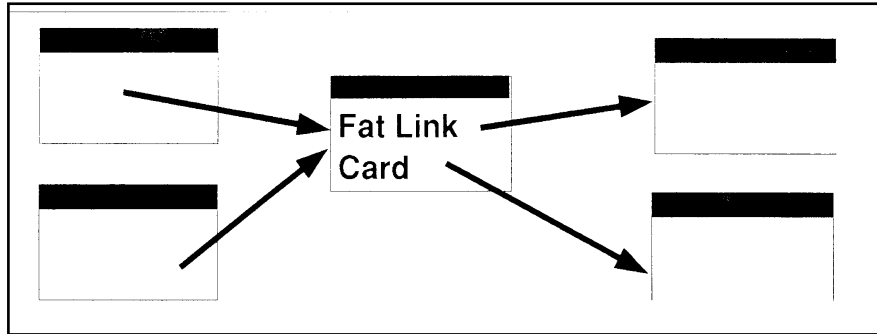
A value of type *SPECIFIER* describes a single end of a link. We include the variable *presentSpec* in the *SPECIFIER* schema so we can model different ways of visually showing links as we follow them (based on the specifier used), as illustrated in the example shown in Figure 2.



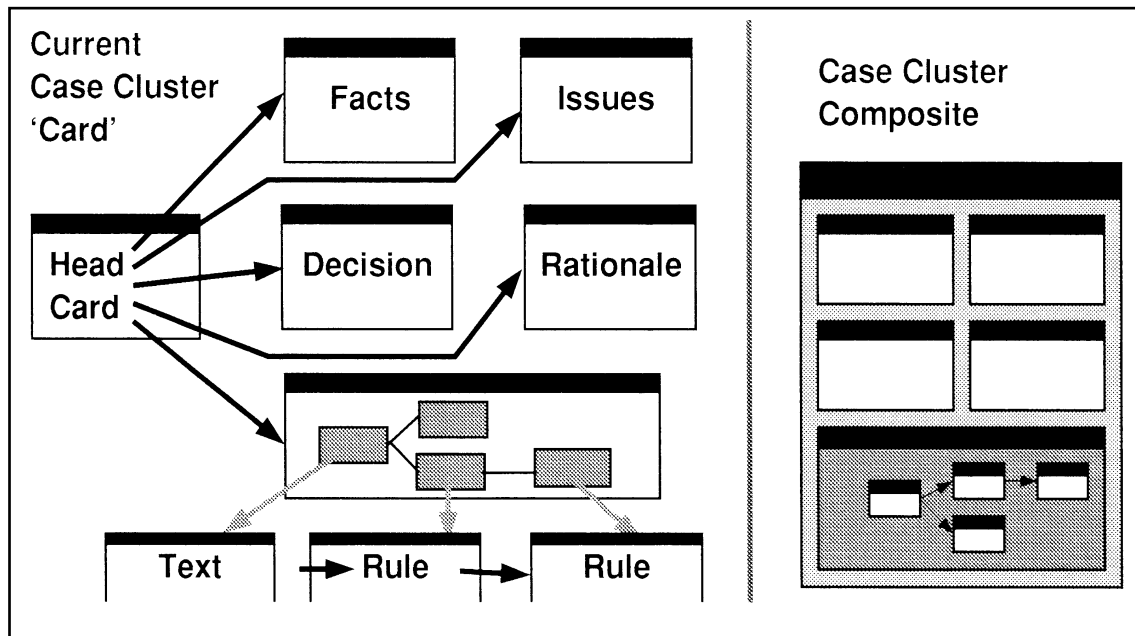
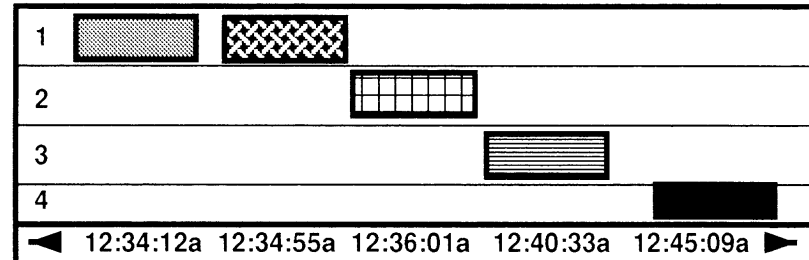
# Dexter Components



# Need for composites (Halasz, 1990)



Example: IGD timeline history display



# Dexter composites

---

We use the recursive type, *BASE\_COMPONENT*, to describe the base components of a hypertext system.

$$\begin{aligned} \text{BASE\_COMPONENT} ::= & \text{atom}\langle\langle \text{ATOM} \rangle\rangle \\ & | \text{link}\langle\langle \text{LINK} \rangle\rangle \\ & | \text{composite}\langle\langle \text{seq } \text{BASE\_COMPONENT} \rangle\rangle \end{aligned}$$

Finally, the schema *COMPONENT* represents a base component along with its associated information.

*COMPONENT*

$\begin{aligned} \text{compBase} : & \text{BASE\_COMPONENT} \\ \text{compInfo} : & \text{COMP\_INFO} \end{aligned}$
---

Components can have sub-components and the same component may be a sub-component to more than one component. This relationship will be denoted by *\_subcomp\_* and is defined below.

$\text{\_subcomp\_} : \text{COMPONENT} \leftrightarrow \text{COMPONENT}$
$\forall c_1, c_2 : \text{COMPONENT} \bullet$ $c_1 \text{ subcomp } c_2 \Leftrightarrow$ $\text{base}(c_1) \in \text{ran}(\text{composite}^\sim(\text{base}(c_2)))$

# STORAGE layer functions

<b>CreateComponent</b>	Creates a new component and adds it to the hypertext. Ensures that the range of the accessor function is extended to include the new component. The resolver function is also extended so that there is at least one specifier for the new component's corresponding UID.
<b>CreateAtomicComponent</b>	Takes an atom and a presentation specification and uses CreateComponent to create a new atomic component.
<b>CreateLinkComponent</b>	Takes a link and a presentation specification and uses CreateComponent to create a new link component.
<b>CreateCompositeComponent</b>	Takes a collection of base components and a presentation specification and uses CreateComponent to create a new composite component.
<b>CreateNewComponent</b>	Invoked from the run-time layer, calls one of CreateAtomicComponent, CreateLinkComponent, or CreateCompositeComponent.
<b>DeleteComponent</b>	Deletes a component ensuring that any links whose specifiers resolve to that component are removed.
<b>ModifyComponent</b>	Modifies a component ensuring that its associated information remains unchanged, that its type (atom, link, or composite) remains unchanged, and that the resulting hypertext remains link consistent.
<b>GetComponent</b>	Takes a UID and uses the accessor function to return a component. If the UID represents a link component, returns either a source or destination specifier for that component.
<b>AttributeValue</b>	Takes a component UID and an attribute and returns the value of the attribute.
<b>SetAttributeValue</b>	Takes a component UID, a value and an attribute, and sets the value of the attribute.
<b>AllAttributes</b>	Returns the set of all component attributes.
<b>LinksToAnchor</b>	Takes an anchor and its containing component, and returns the set of links that refer to the anchor.
<b>LinksTo</b>	Takes a hypertext and a component UID, and returns the UIDs of links resolving to that component.

## Behavior of DeleteComponent (Halasz & Schwartz, 1990, p. 115)

---

### 2.4 Deleting A Component

In deleting a component we must ensure that we remove any links whose specifiers resolves to that component.

$$\underline{\text{DeleteComponent}} : \text{HYPERTEXT} \times \text{UID} \rightarrow \text{HYPERTEXT}$$
$$\begin{aligned} \text{DeleteComponent} = & (\lambda H : \text{HYPERTEXT}; \text{uid} : \text{UID} \bullet \\ & (\mu H' : \text{HYPERTEXT} \mid \exists \text{uids} : \mathbf{F} \text{UID} \mid \\ & \quad \text{uids} = \{\text{uid}\} \cup \text{linksTo}(H, \text{uid}) \bullet \\ & \quad H'.\text{components} = H.\text{components} \setminus H.\text{accessor}(\text{uids}) \wedge \\ & \quad H'.\text{accessor} = \text{uids} \triangleleft H.\text{accessor} \wedge \\ & \quad H'.\text{resolver} = H.\text{resolver} \triangleright \text{uids})) \end{aligned}$$

## At least two specifiers per link (Halasz & Schwartz, 1990, p. 106-107)

---

Links must include at least two specifiers. What appear to be one-way links, such as Hypercard buttons, can be modeled as two-way links with the button end having a *DIRECTION* with value *NONE* and the other end having a *DIRECTION* with value *TO*. The two *specifiers* link constraint simplifies the hypertext model. On the other hand there is no reason not to have multi-way links, and so the model accomodates them. In the most general model, duplicate specifiers are allowed. The only constraint is that at least one specifier have a direction of *TO*.

*LINK*

*specifiers* : seq *SPECIFIER*

$\#specifiers \geq 2$

$\exists s : \text{ran } specifiers \bullet s.direction = TO$



# Ensuring link "consistency" (Halasz & Schwartz, 1990, p. 113-114)

---

$\text{linkConsistent\_} : P \text{ HYPertext}$

$\forall H : \text{HYPertext} \bullet$

$\text{linkConsistent } H \Leftrightarrow$

$(\forall l : \text{LINK}; s : \text{SPECIFIER} |$

$(\exists cl : \text{LinkComp} | cl \in H.\text{components} \bullet$

$l = \text{link}^{\sim}(\text{base}(cl))) \wedge$

$s \in \text{ran } l.\text{specifiers} \bullet$

$(\exists c : \text{COMPONENT} | c \in H.\text{components} \bullet$

$(H.\text{accessor} \circ H.\text{resolver})(s.\text{componentSpec}) = c))$

Creating a new link component is then given by the following function.

$\text{createLinkComponent} : \text{HYPertext} \times \text{LINK} \times \text{PRESENT\_SPEC}$   
 $\rightarrow \text{HYPertext} \times \text{COMPONENT}$

$\forall H : \text{HYPertext}; l : \text{LINK}; ps : \text{PRESENT\_SPEC} \bullet$

$\exists H' : \text{HYPertext}; c : \text{COMPONENT} |$

$c = \text{component}(\text{link}(l), \text{minInfo}(ps)) \wedge$

$H' = \text{createComponent}(H, c) \wedge$

$\text{createLinkComponent}(H, l, ps) = (H', c) \bullet$

$\text{linkConsistent } H'$

# RUNTIME layer operations

---

<b>openSession</b>	Creates a session for a given hypertext. A session begins with no instantions.
<b>openComponents</b>	Opens a set of new instantiations on a given set of components.
<b>presentComponent</b>	Takes a specifier and a presentation specification and creates an instantiation for the associated component.
<b>followLink</b>	Uses openComponents to present any components referred to by the "TO" specifiers of any links with anchors represented by a given link marker.
<b>newComponent</b>	Opens a new instantiation on a newly created component.
<b>unPresent</b>	Removes an instantiation.
<b>editInstantiation</b>	Instantiations can be modified by editing them. A call to realizeEdits is required to save the changes.
<b>realizeEdits</b>	Saves an instantiation's editing changes to the corresponding component by calling ModifyComponent.
<b>deleteComponent</b>	Deletes the component associated with a given instantiation. Also removes any other instantiations for that component.
<b>closeSession</b>	Closes a given session. Note that by default, pending changes to instantiations are not saved.

# A simple Dexter Interchange Format

---

```
<hypertext>
  <component>
    <type> text </type>
    <uid> 21 </uid>
    <data> This is some text .... </data>
    <anchor>
      <id> 1 </id>
      <location> d13 </location>
    </anchor>
  </component>
  <component>
    <type> text </type>
    <uid> 777 </uid>
    <data> This is some other text .... </data>
    <anchor>
      <id> 1 </id>
      <location> 13-19 </location>
    </anchor>
  </component>
  <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>
```

---

# Problems in Dexter

# Dexter problems revealed through practical use

---

- Although not meant as a design spec we took Dexter as a starting point for the design of the DEVISE Hypermedia (DHM) framework.
- Dexter concepts were captured in an object oriented framework.
- Originally, we aimed at achieving hypermedia structure and behavior according to the spec.

# Dexter worked as a design spec, BUT...

---

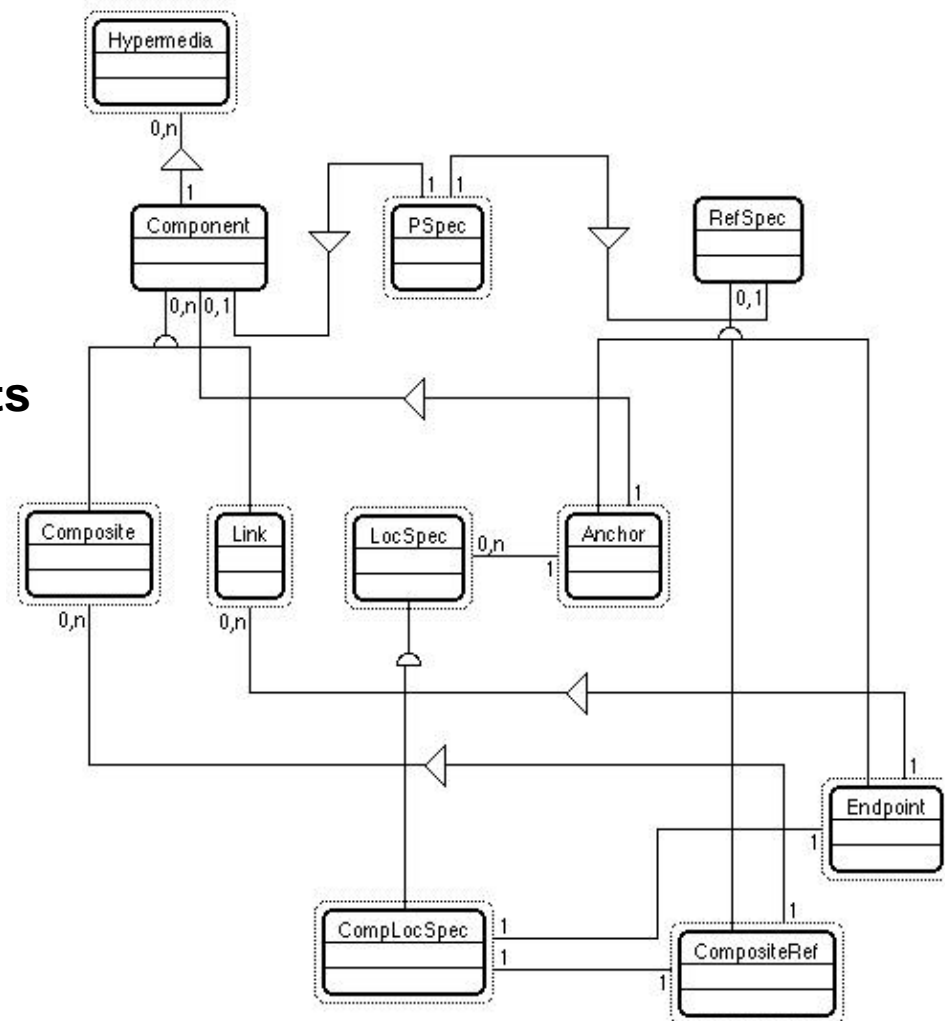
...clarifications and extensions were needed in several cases, e.g.:

- ***Dangling links:*** We believe not only in allowing dangling links, but in actively supporting them in a variety of situations.
- ***Link directionality:*** Are the direction attributes TO, FROM, BIDIRECT, NONE adequate? Which senses of link directionality are they meant to cover?
- ***Anchors.*** Is one anchor type sufficient? What do specifiers point at for "whole-component-links"? Are anchors shared between links?
- ***Components:*** How do we connect components to their contents in an integrated hypermedia system that doesn't "own" all material?
- ***Composites:*** Dexter composites only model the internal structure of *data objects*. But composites should also be used to model structures built from components (e.g. tabletops, browsers, query results)!
  
- ***CSCW:*** Dexter is silent regarding multiuser aspects and distribution.
- ***Multimedia and time:*** Nor does Dexter handle temporal issues.

# Our extended object oriented Dexter-based model for open hypermedia

The objective is to:

- provide a unified model for anchors and embedded references
- provide modelling power to better express dynamic aspects of hypermedia
- extend Dexter into a better model for open hypermedia



# Why not dangling links?

---

## Four reasonable and foreseeable dangling situations:

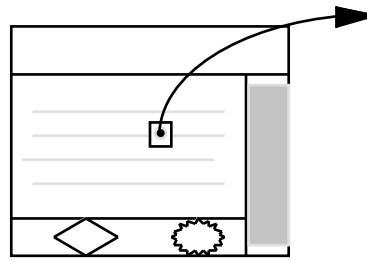
1. deleted endpoint component,
2. deleted endpoint anchor,
3. unavailable data objects in endpoint component,
4. invalid anchor value due to editing outside the hypermedia.

**Intentionally incomplete links (open for later addition of endpoints) should also be supported.**

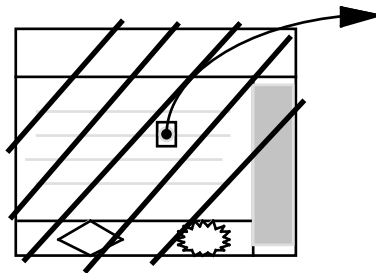


# Dangling link examples

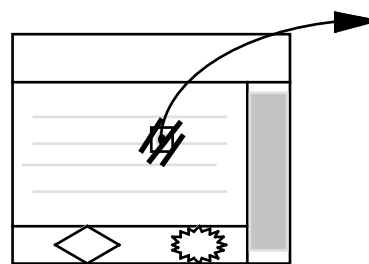
---



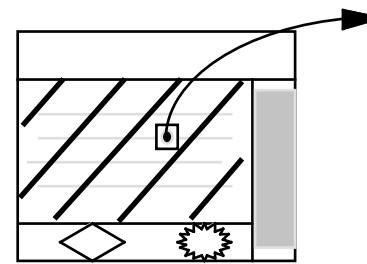
Normal link endpoint



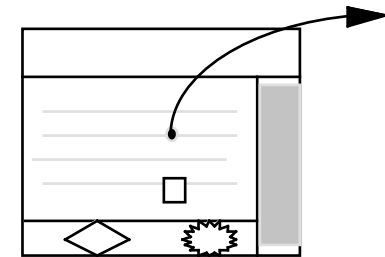
Deleted endpoint component



Deleted endpoint anchor

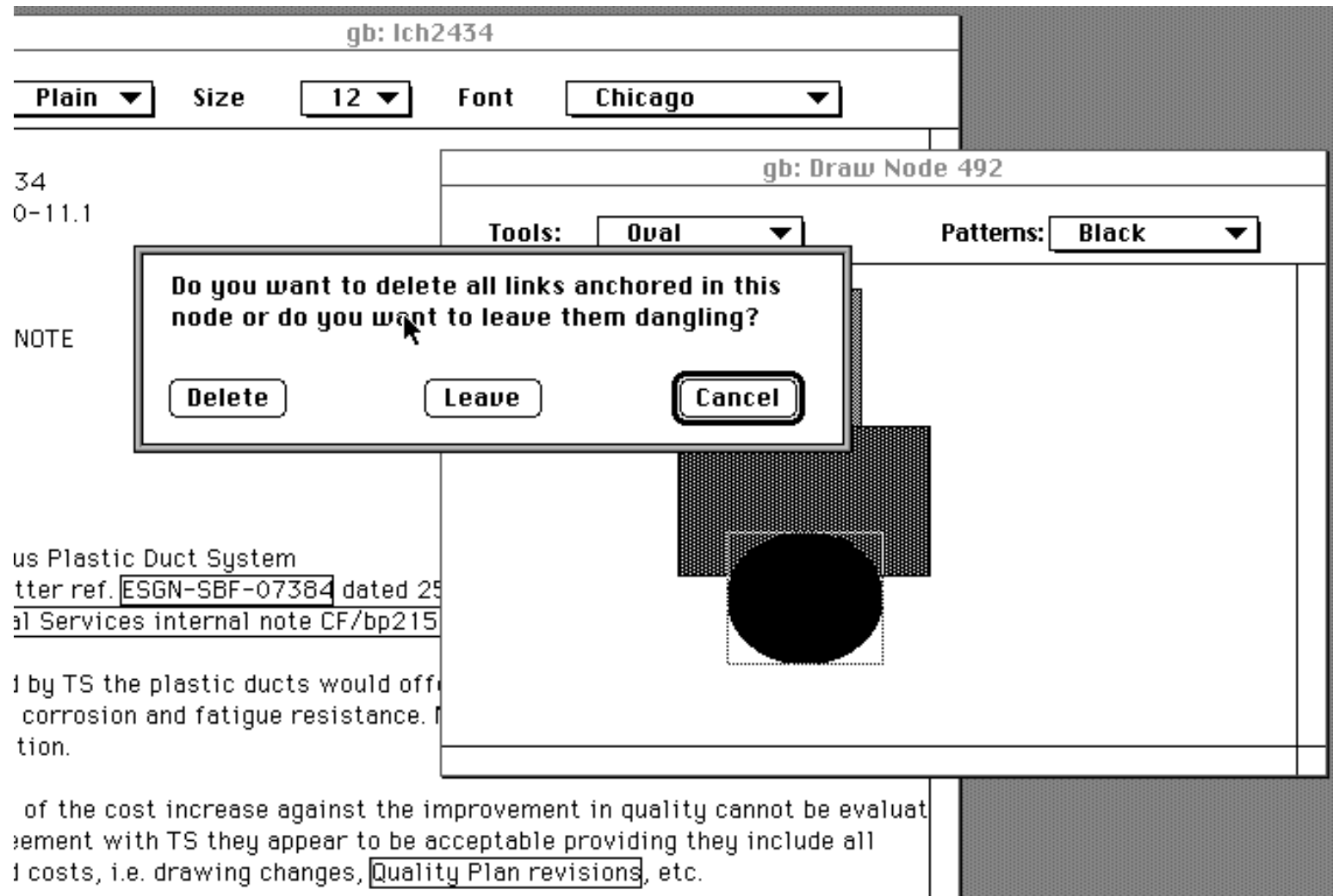


Unavailable data objects



Invalid anchor value

# Leaving dangling links when deleting components



# Options when following a link with a dangling endpoint

J. No. 01.20-11.1

Page 1

INTERNAL NOTE

To : LML  
From : AGi

cc : EKr

USL PT-Plus Plastic Duct System

Re.: ESG letter ref. [ESGN-SBF-07384](#) dated 25 Jun 92

& [Technical Services internal note CF/bp2159](#) dated 06 Aug 92

As advised by TS the plastic ducts would offer a small "in service" improvement with regards to corrosion and fatigue resistance. No advantages would be gained in prefabrication.

Evaluation of the cost increase against the improvement in quality cannot be evaluated but in agreement with TS they appear to be acceptable providing they include all associated costs, i.e. drawing changes, [Quality Plan revisions](#), etc.

**This link has a dangling endpoint! Do you want to delete the link, delete the endpoint, or to enable re-linking to another destination?**

Delete Link

Delete Endpoint

Enable

Cancel

# Following an incomplete link

4 dated 25 Jun 92  
CF/bp2159 dated 06 Aug 92

would offer a small "in se  
sistance. No advantages w

inst the improvement in qu  
ar to be acceptable provid  
ges, Quality Plan revisions

nt for the bridge it is recd  
astic ducts.

The screenshot shows a software interface with a window titled "gb: Link489". Inside this window, there is a list box containing the text "Ich2434". To the right of the list box are five buttons: "Refresh", "Present", "Edit anchor", "Make current", and "Remove spec". The "Present" button is highlighted with a double border. In the foreground, there is a dialog box with a speech bubble icon. The dialog box contains the text: "This is a link with only one endpoint! Use Add Endpoint command to add more endpoints!". At the bottom right of the dialog box is an "OK" button.

# Different notions of link directionality

---

## **Semantic direction:**

Ordering implied by the semantic relationship between the connected components.

Example: A “supports” link connecting two components is “read” in a certain direction: the argument in A “supports” the claim in B.

## **Creation direction:**

The order in which the link endpoints were created.

Usually the first endpoint created is considered the source of the link, while the last is considered the destination.

## **Traversal direction:**

How the link can be traversed.

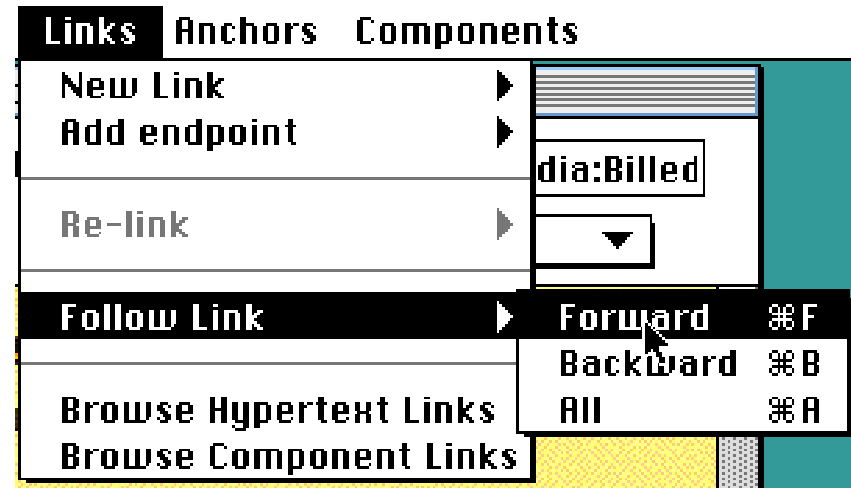
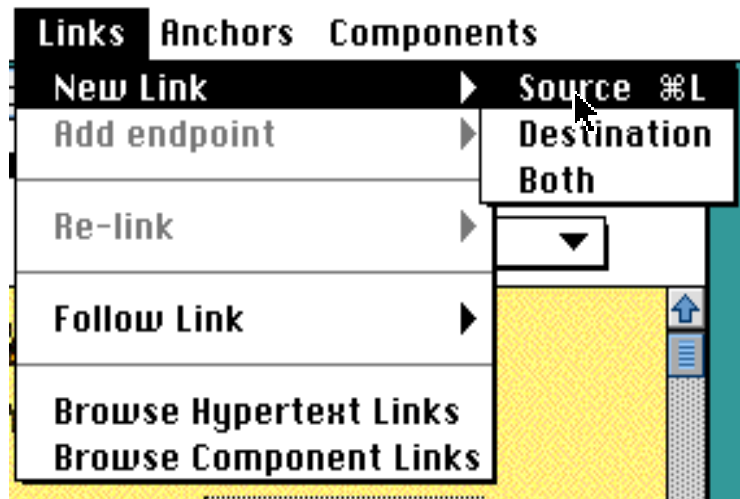
Examples: WWW links can only be traversed from source to destination.

NoteCards links can be traversed in both directions, although the interface style is different. DHM links can be traversed symmetrically in both directions.

*What about multi-headed links?*

# Link Directionality in DHM

---



# RefSpecs/LocSpecs and directionality

---

Replace directionality constants with link endpoint “types”: SOURCE, DESTINATION, BOTH

Enable tailoring of new endpoint types, e.g. HIDDEN.

Solve the “NONE problem” by disentangling the notions of embeddedness and directionality:

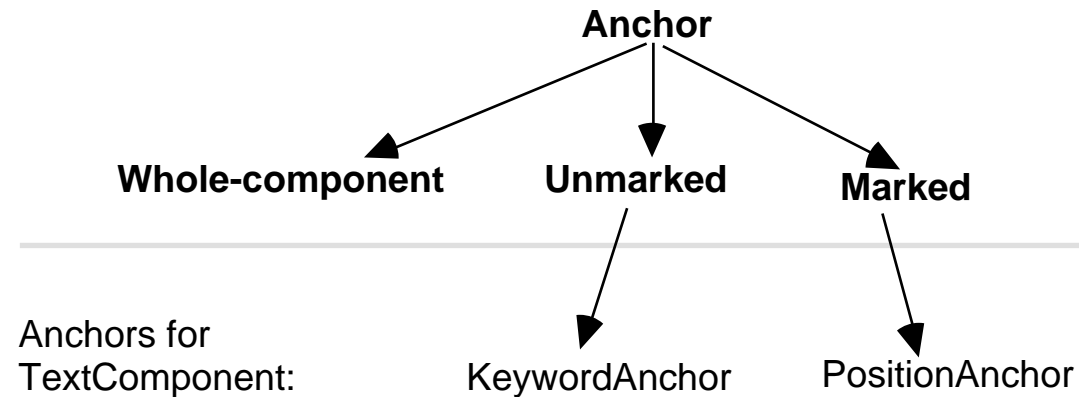
- *location specification*: free-floating description of a hypermedia location
- *reference specification*: location specification packaged as an object within the hypermedia system

(For details, see our HT'96 paper.)

# Anchors in a Dexter-based hypermedia

---

## Anchor classes in DHM:



- Locating an unmarked anchor in the contents of a component requires computation.
- Marked anchors are located by their linkMarkers.
- Anchors are reused if possible, e.g. there is only one Whole-component anchor per component.



# Example: linking and anchoring in html

---

## Linking to an entire page:

- Syntax: `http://www.daimi.aau.dk/~kgronbak/KGronbakHome.html`

## Linking to a predefined anchor in a page:

- Syntax: `http://www.daimi.aau.dk/~kgronbak/KGronbakHome.html#hypermedia`
- There must be an anchor by the given name defined in the destination node:

```
<A NAME="hypermedia">Hypermedia stuff</a>
```

## “Computed” link that triggers a search:

- Syntax: `http://cernvm/FIND/?sgml+cms`
- The address must be for a node which is marked as an INDEX
- The node’s server must support search.
- Usually, the result is a new computed document, with links to search hits.

# RefSpecs/LocSpecs and Anchoring

---

## **Anchors with “redundant” specifications:**

- “marked” attributes, e.g. ID of an object
- “unmarked” attributes, e.g. search query

## ***Transient* anchors:**

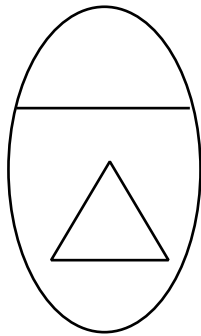
- Constructed at run-time
- Usually based on a selection in an editor
- Typically exist for the duration of a link traversal

**(For details, see our HT'96 paper.)**

# Integration and component contents

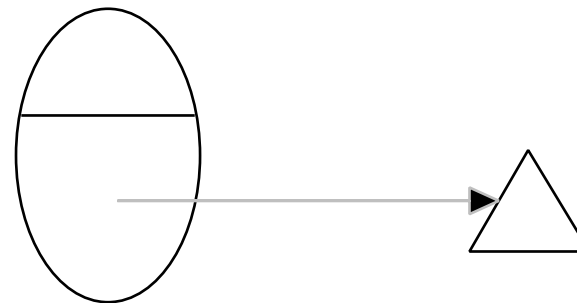
---

**Component contents are either embedded or stored externally.**



**Example:**

**An internal drawing editor that stores drawings as part of the component in the OODB.**

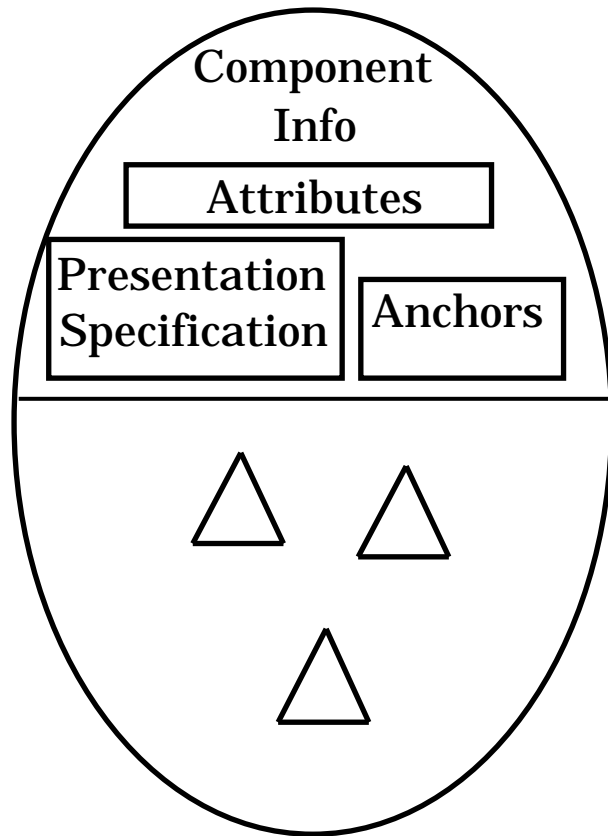


**Example:**

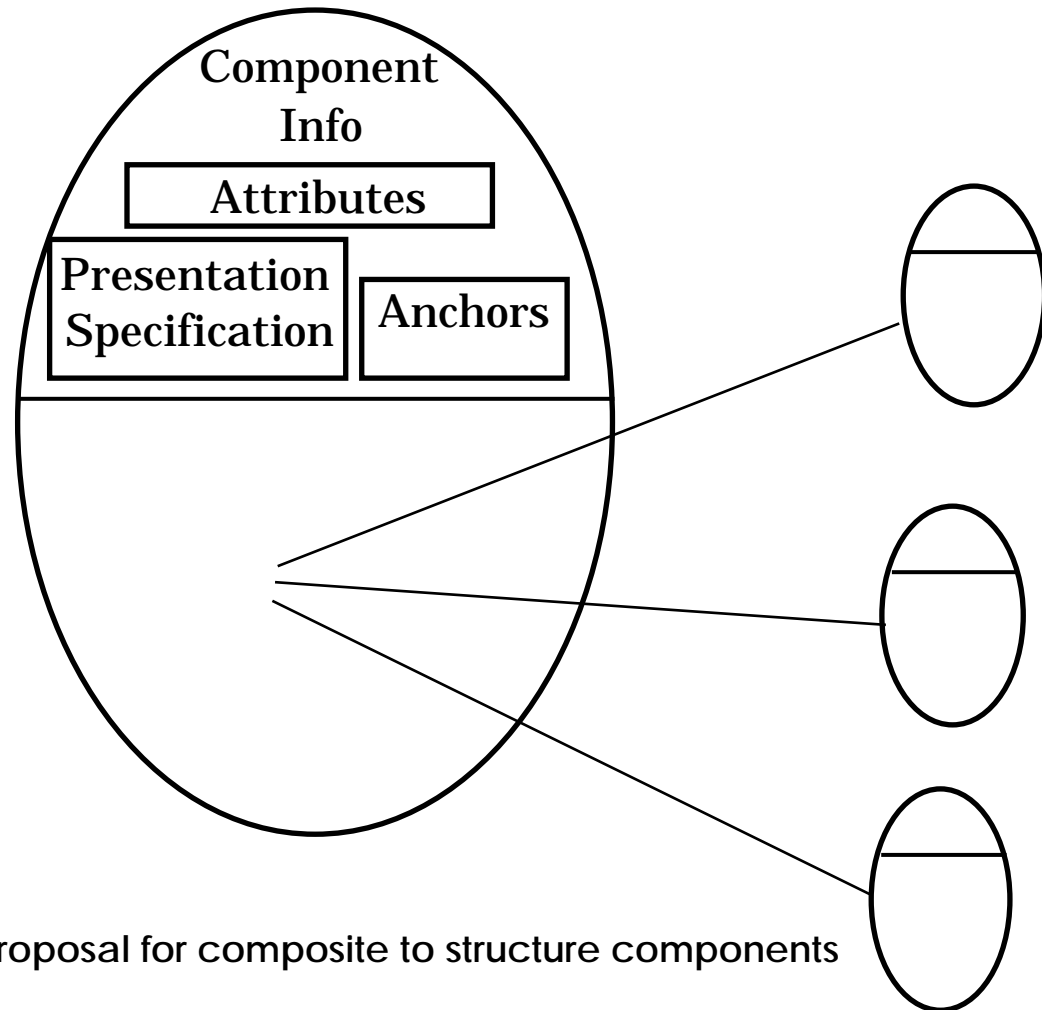
**A video component whose video data object is stored in a separate file.**

# Need for composites to structure other components

---



Dexter Composite



Proposal for composite to structure components

# Uses of Composites

---

- **Data objects with internal structure**
  - e.g. video
- **Hierarchical structure**
  - by reference
  - by inclusion
- **Browsers and Queries**
  - virtual
  - computed
- **Paths**
  - GuidedTour
  - TableTop
- **History**
  - data structure: e.g. changes to components, anchors
  - behavior: e.g. list of recent components visited, ...

---

# **Object oriented hypermedia design based on Dexter**

# Object oriented design: Our approach

---

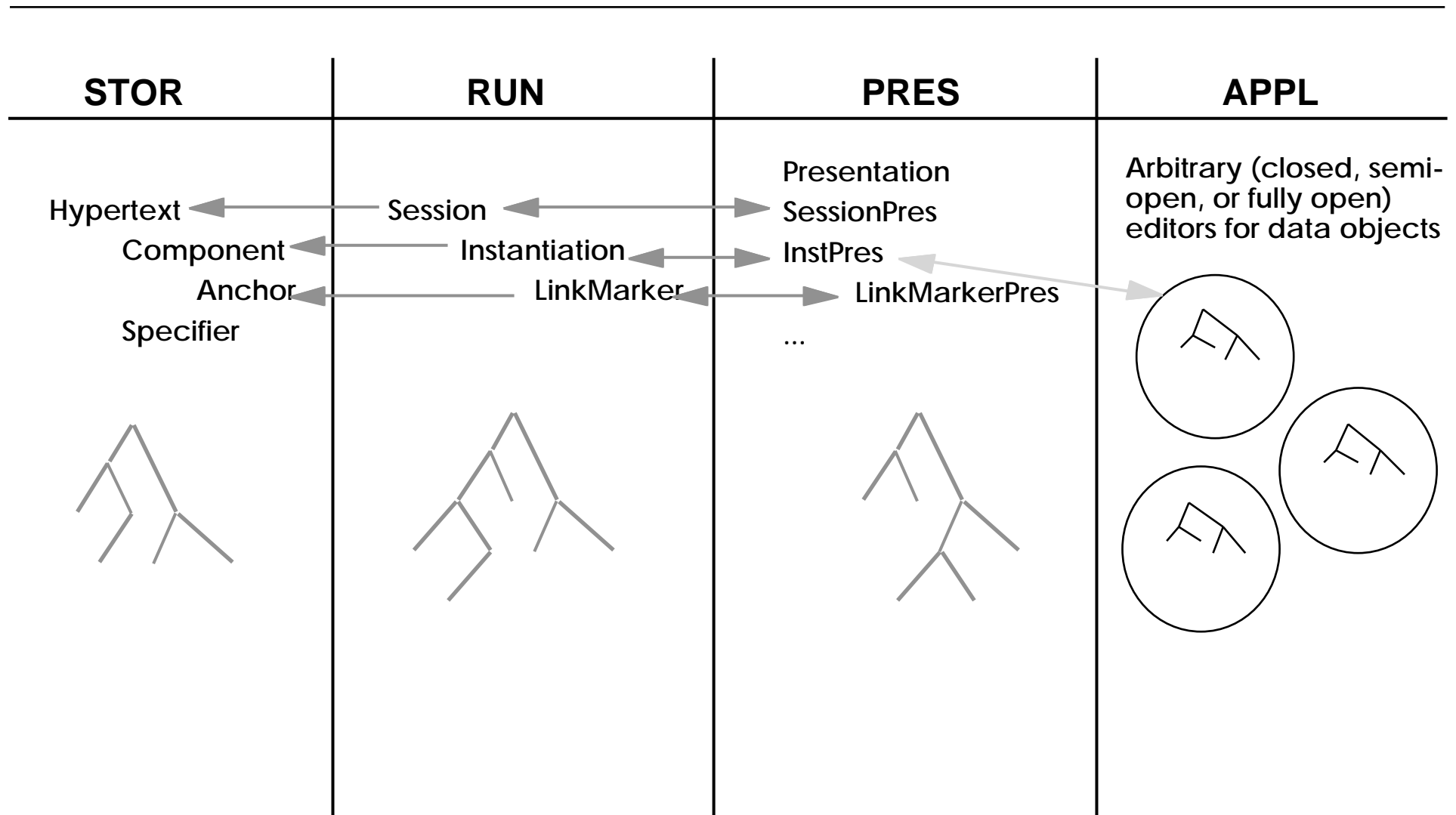
Use lessons learned from the Dexter model as starting points:

- Separation between Storage (persistent) and Runtime (transient)
- Bi-directional links
- Multi-headed (n-ary) links
- A basic notion of Composites
- An interchange format

Develop an object oriented design with:

- generic classes for all the Dexter concepts
- classes organized in an extensible and tailorable framework for hypermedia development
- a persistent object store/OODB to handle Storage layer objects
  - persistence frees us from tedious management of unique IDs, at least locally

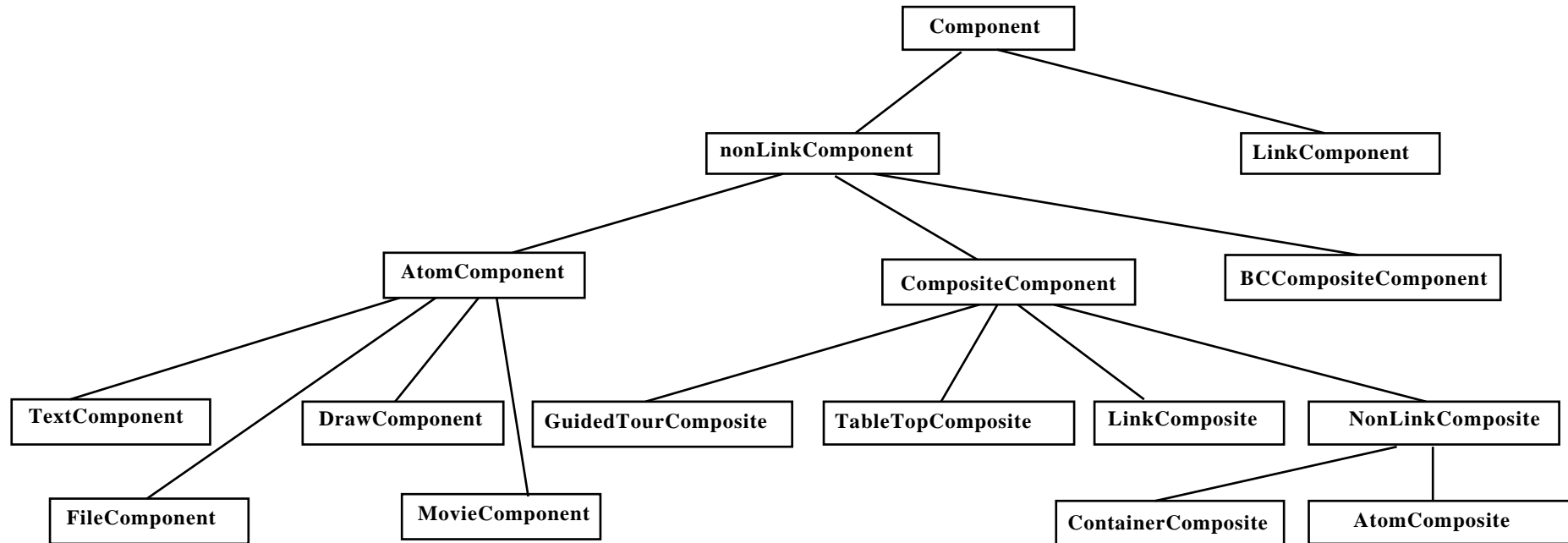
# The object oriented DHM Framework





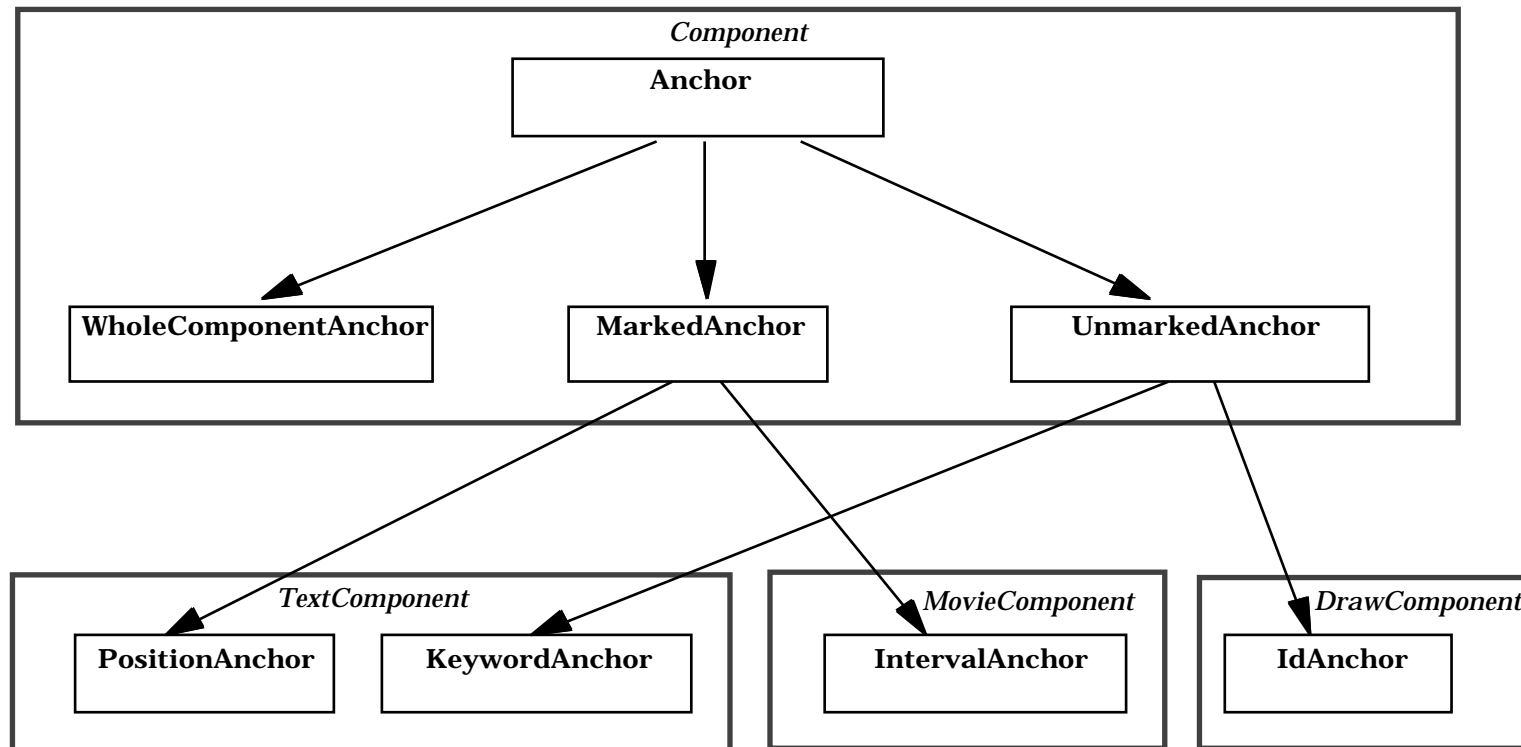
# Storage Layer example (Components)

---

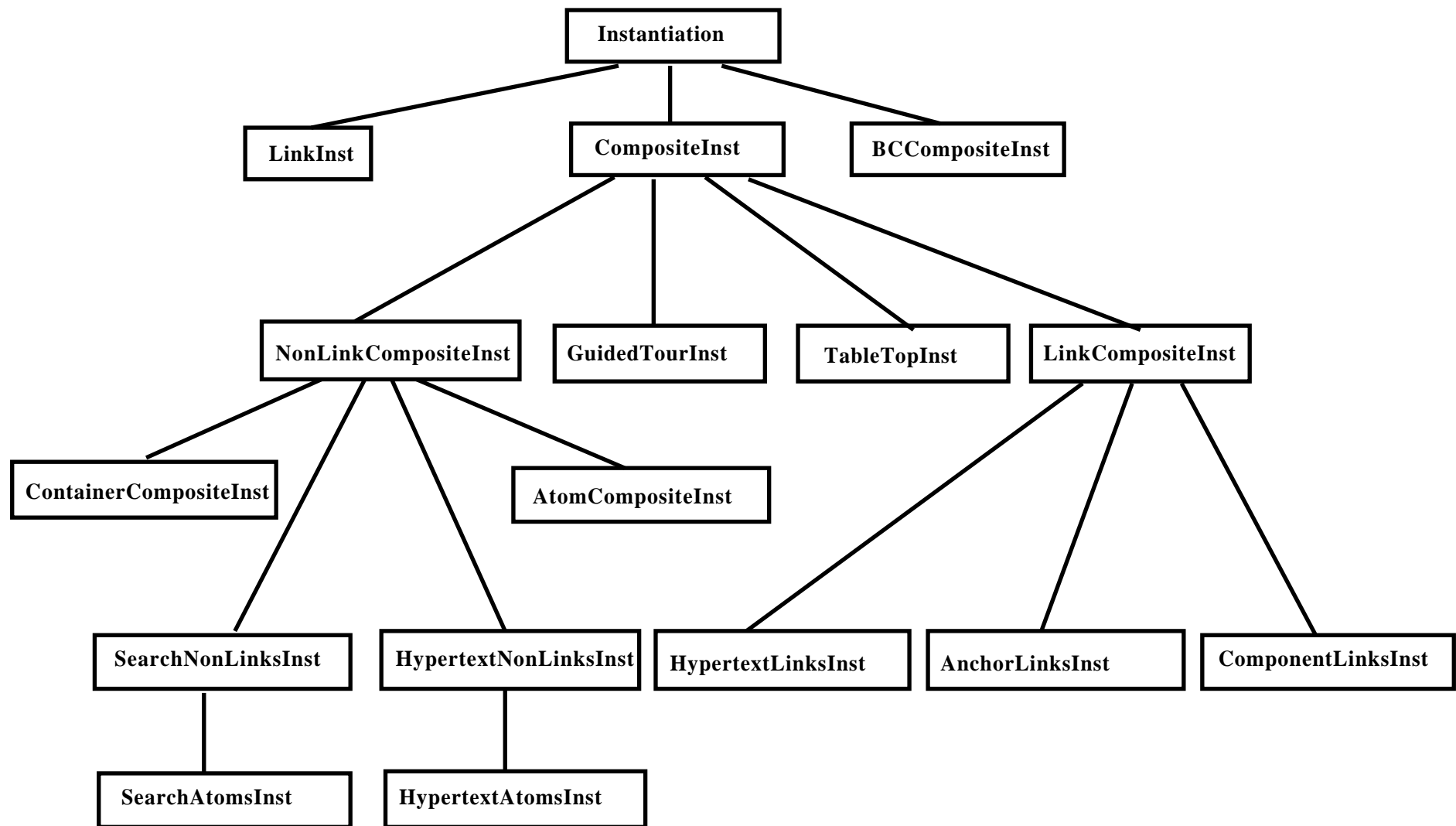


# Storage Layer example (Anchors)

---

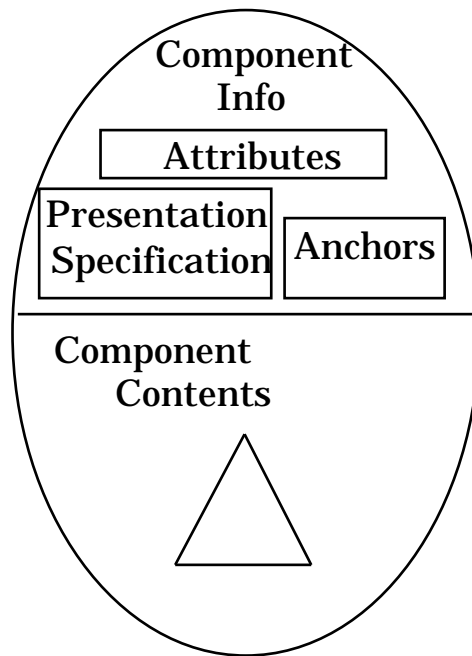


# Instantiation hierarchy

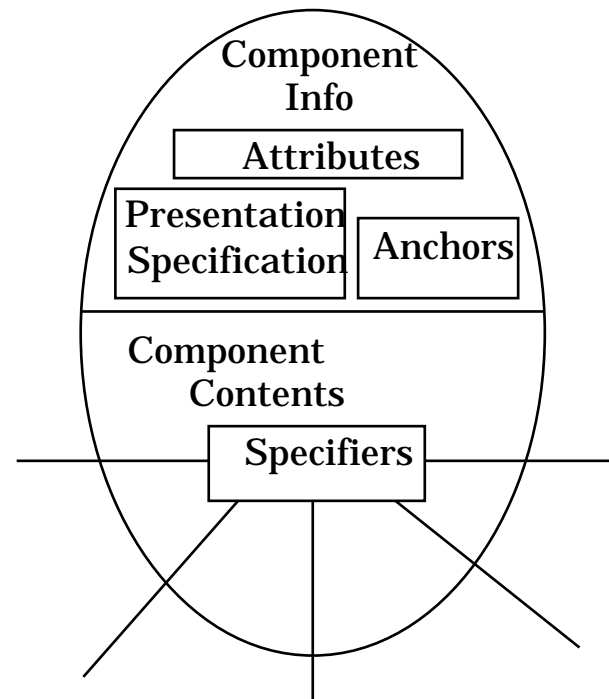


# Dexter Storage: Components

---

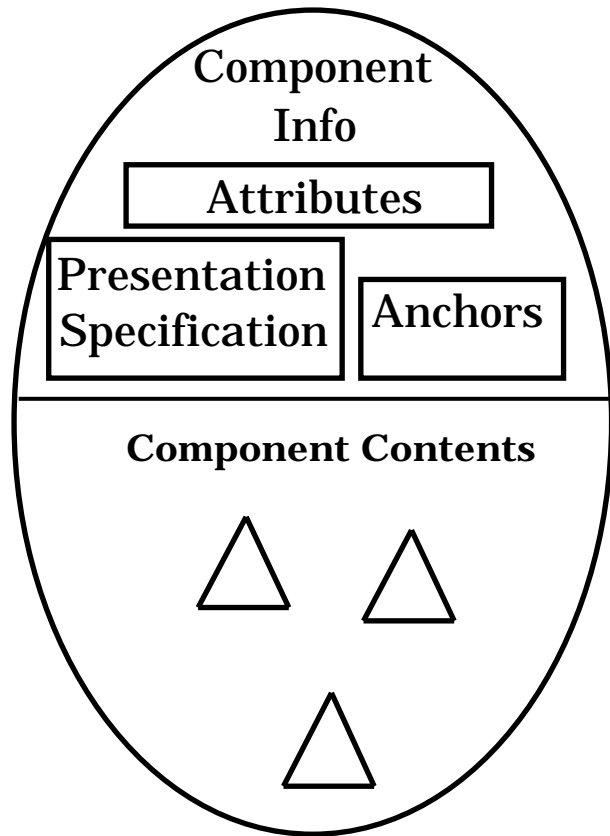


AtomComponent

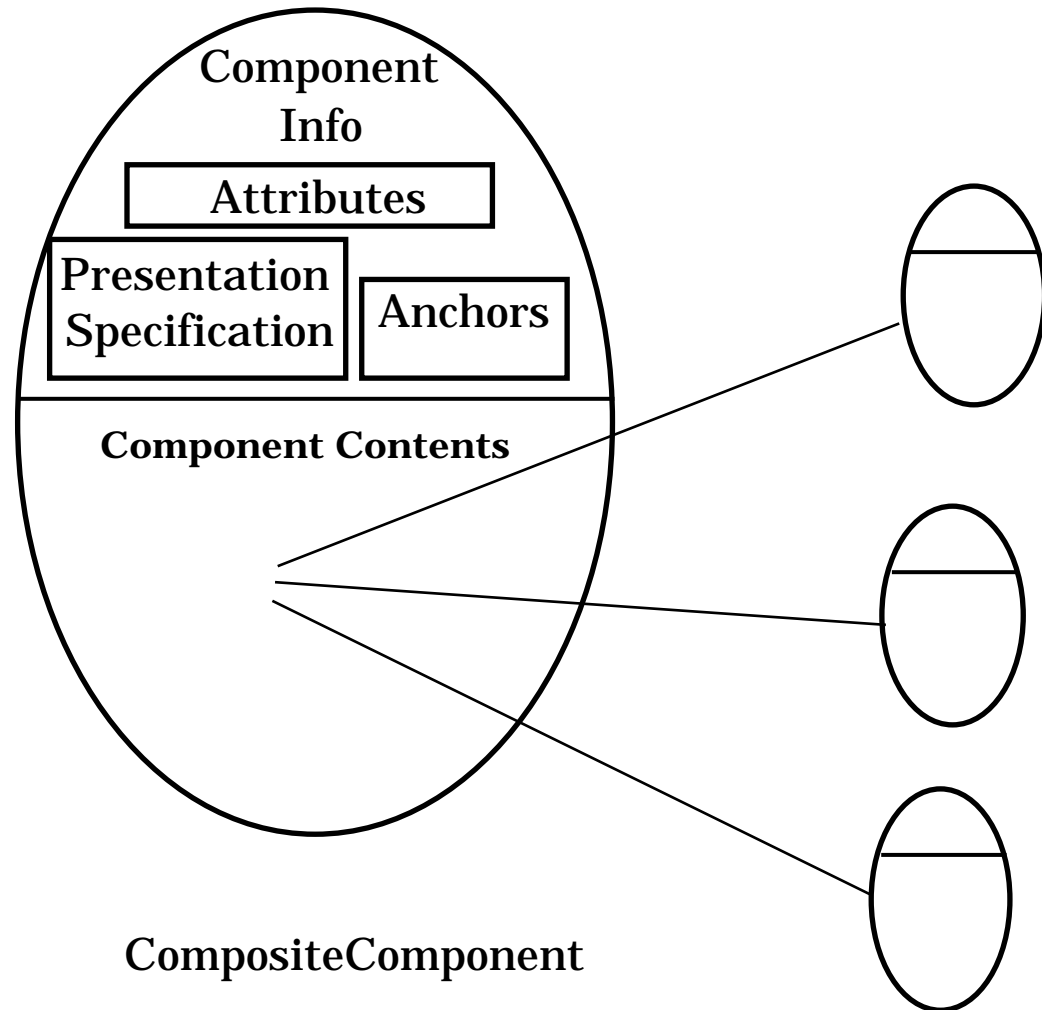


LinkComponent

# DHM general composites



BCCompositeComponent



CompositeComponent

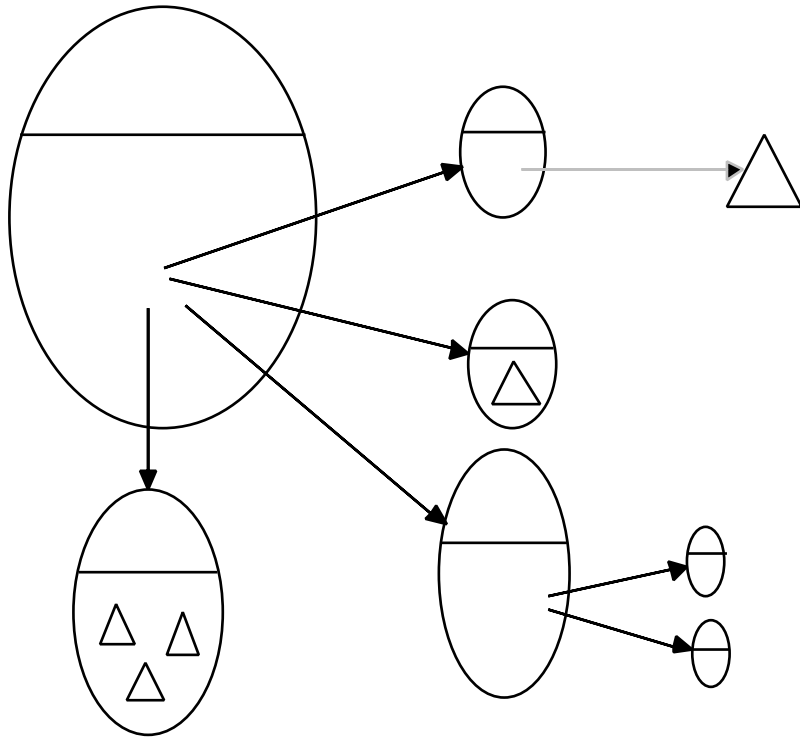
# Aspects of composite contents

---

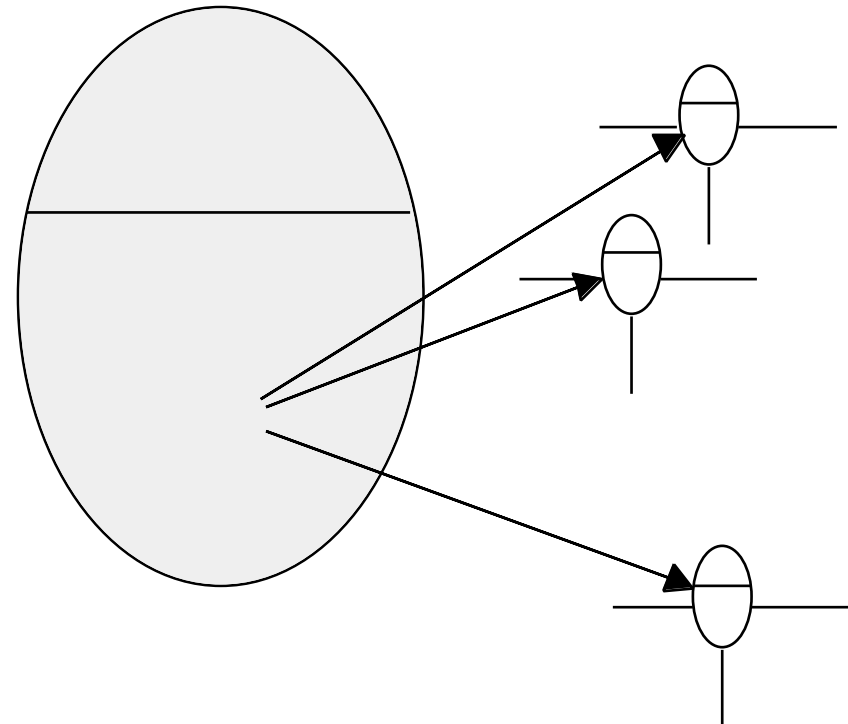
<b>Structure</b>	<b>Type</b>	<b>Definition</b>	<b>Location</b>
<ul style="list-style-type: none"><li>• Unstructured collection</li><li>• Structured collection:<ul style="list-style-type: none"><li>- sorted list</li><li>- keyed table</li><li>- tree</li><li>- ...</li></ul></li></ul>	<ul style="list-style-type: none"><li>• Data objects</li><li>• Components<ul style="list-style-type: none"><li>- restricted types</li><li>- unrestricted</li></ul></li></ul>	<ul style="list-style-type: none"><li>• Encapsulated in this composite</li><li>• Globally visible</li></ul>	<ul style="list-style-type: none"><li>• within composite (inclusion)</li><li>• outside composite (reference)</li></ul>

# Composites containing/referencing components

---

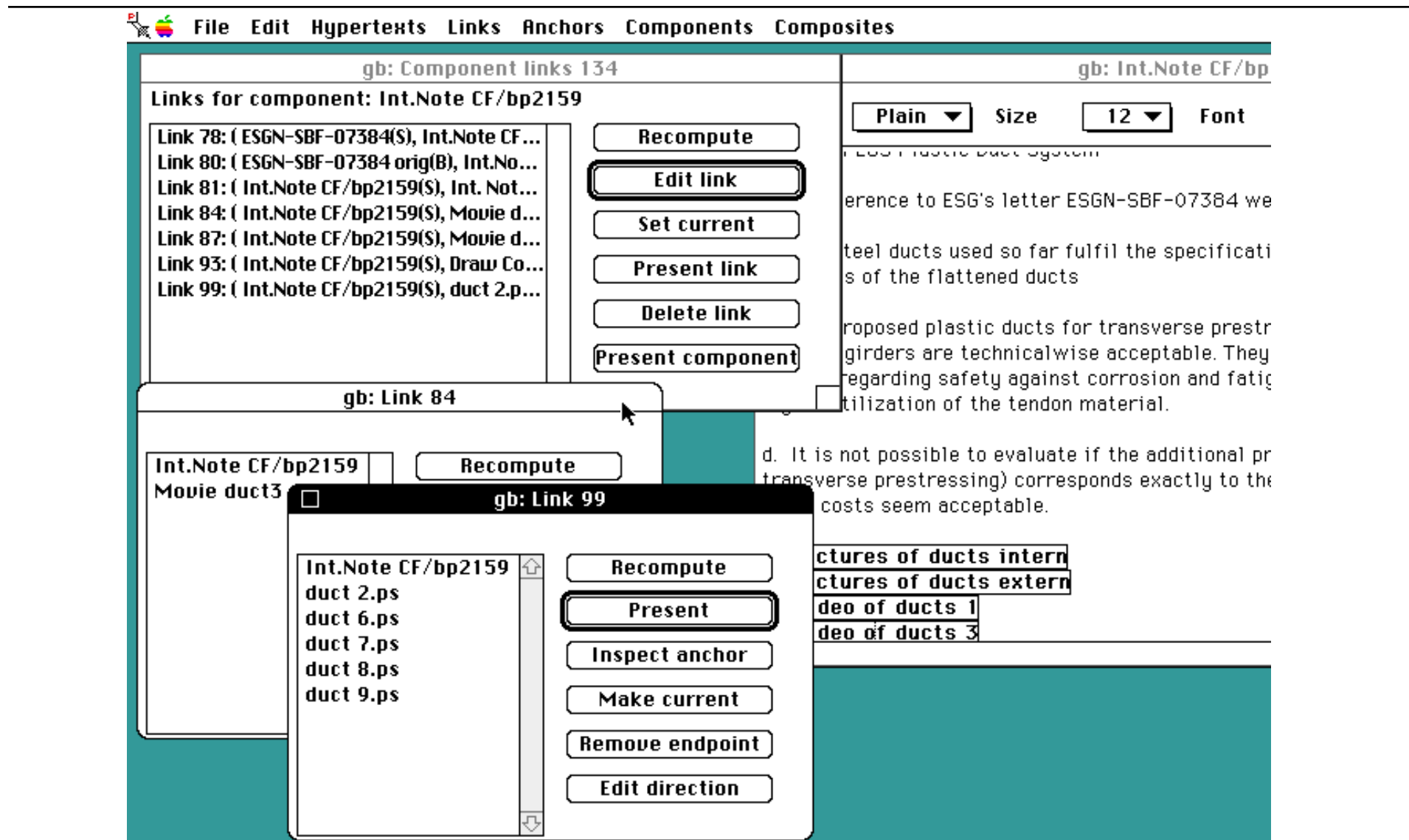


**A TableTopComposite refers to components of arbitrary type.**



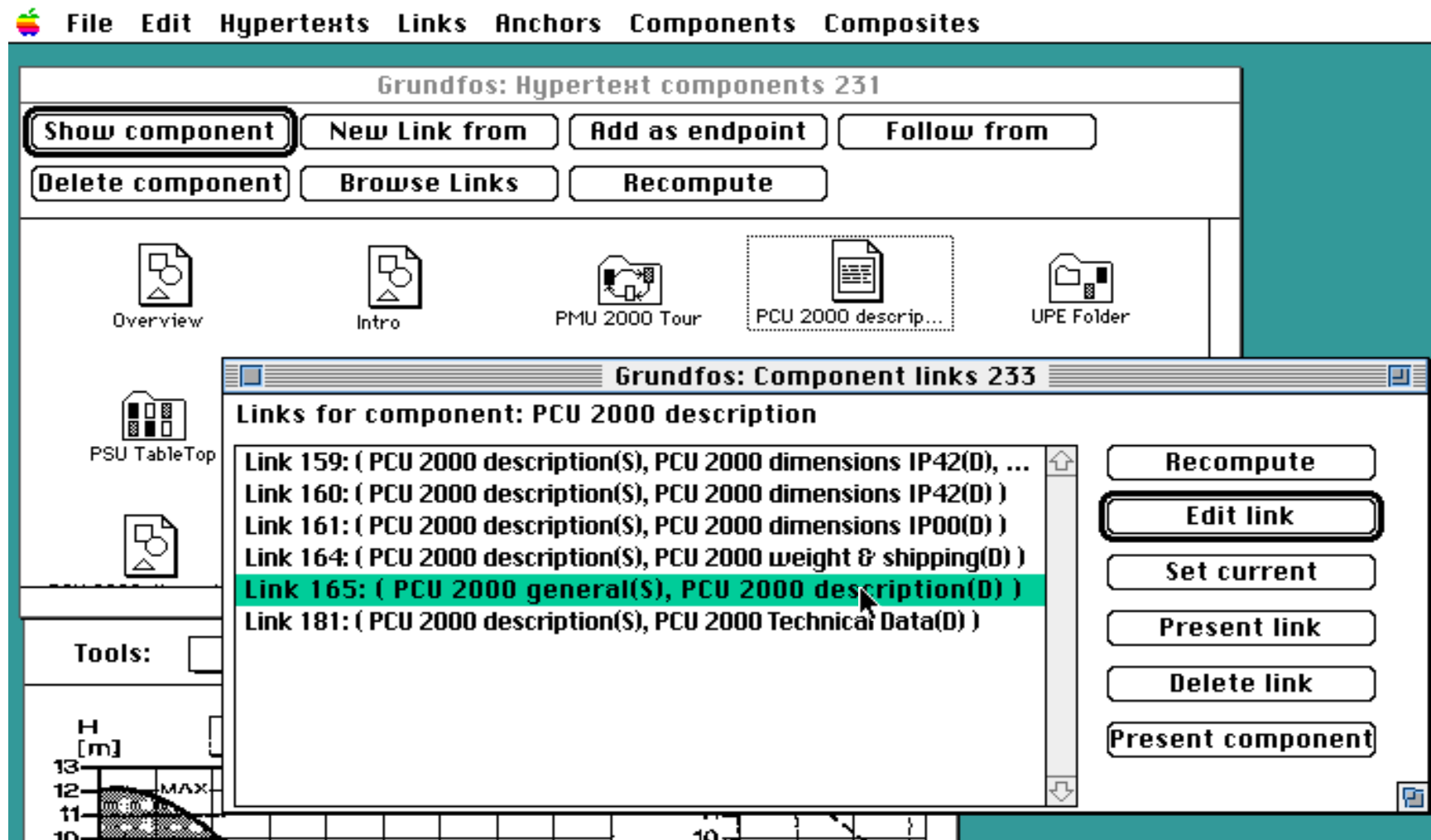
**A virtual LinkComposite refers only to LinkComponents.**

# Example of a virtual link composite





# Browser interfaces



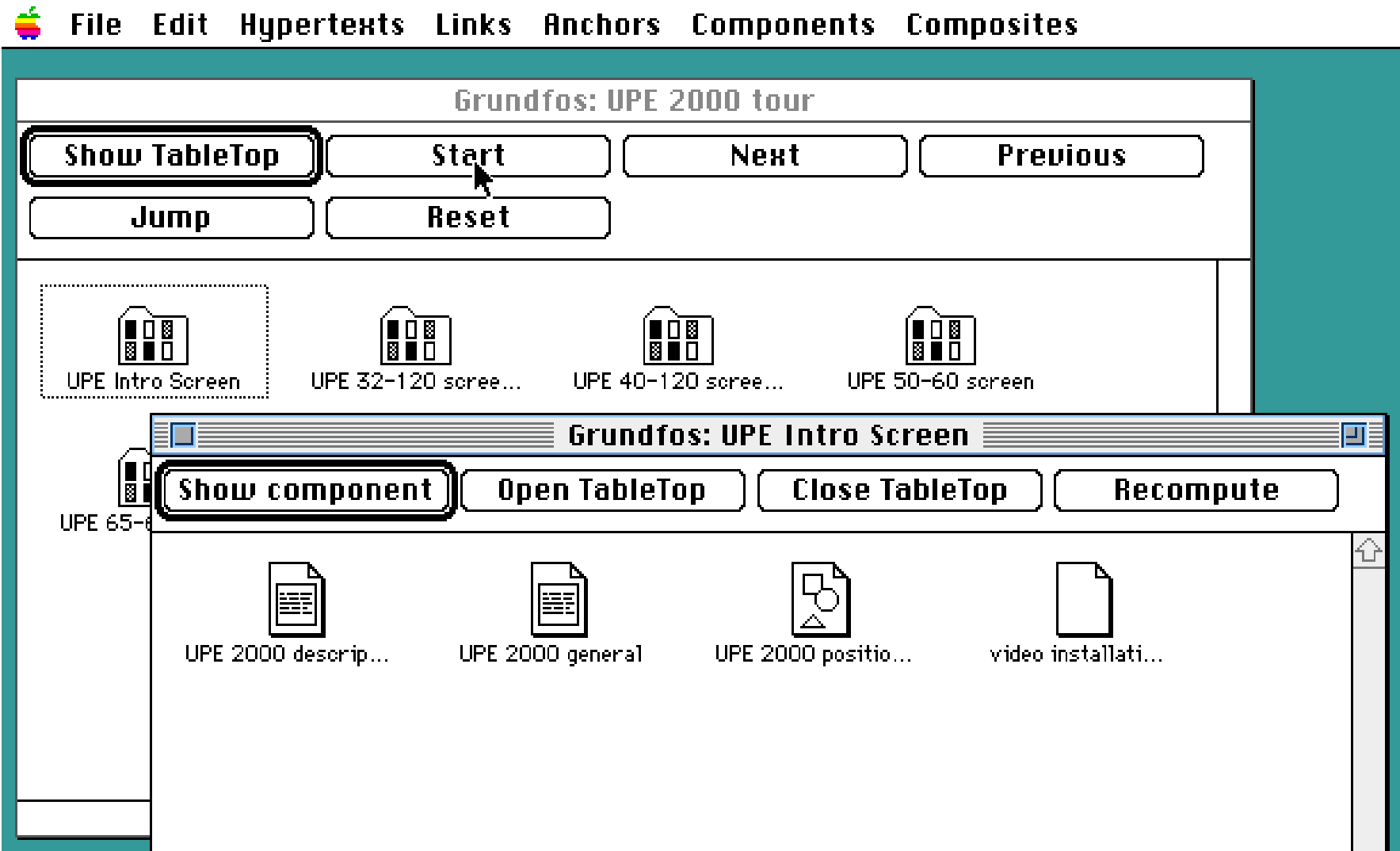
# Representing paths

---

- **GuidedTours and TableTops (Trigg 1988)**
  - descended from Bush's notion of trails
  
- **TableTop**
  - Captures a configuration of component presentations on the screen
  - Opens and closes the configuration as a unit
  - Can be computed or built manually
  
- **GuidedTour**
  - A (branching) sequence of TableTops
  - Behavior includes, for example, the 'Next' operation which closes current TableTop and opens the next in the sequence
  - Feedback is given to show which part of the tour has been visited

*These concepts can be implemented by means of composites, but it raises a need for Pspecs on composite "pointers."*

# DHM GuidedTour and TableTop



---

# **Dexter-based Architecture**

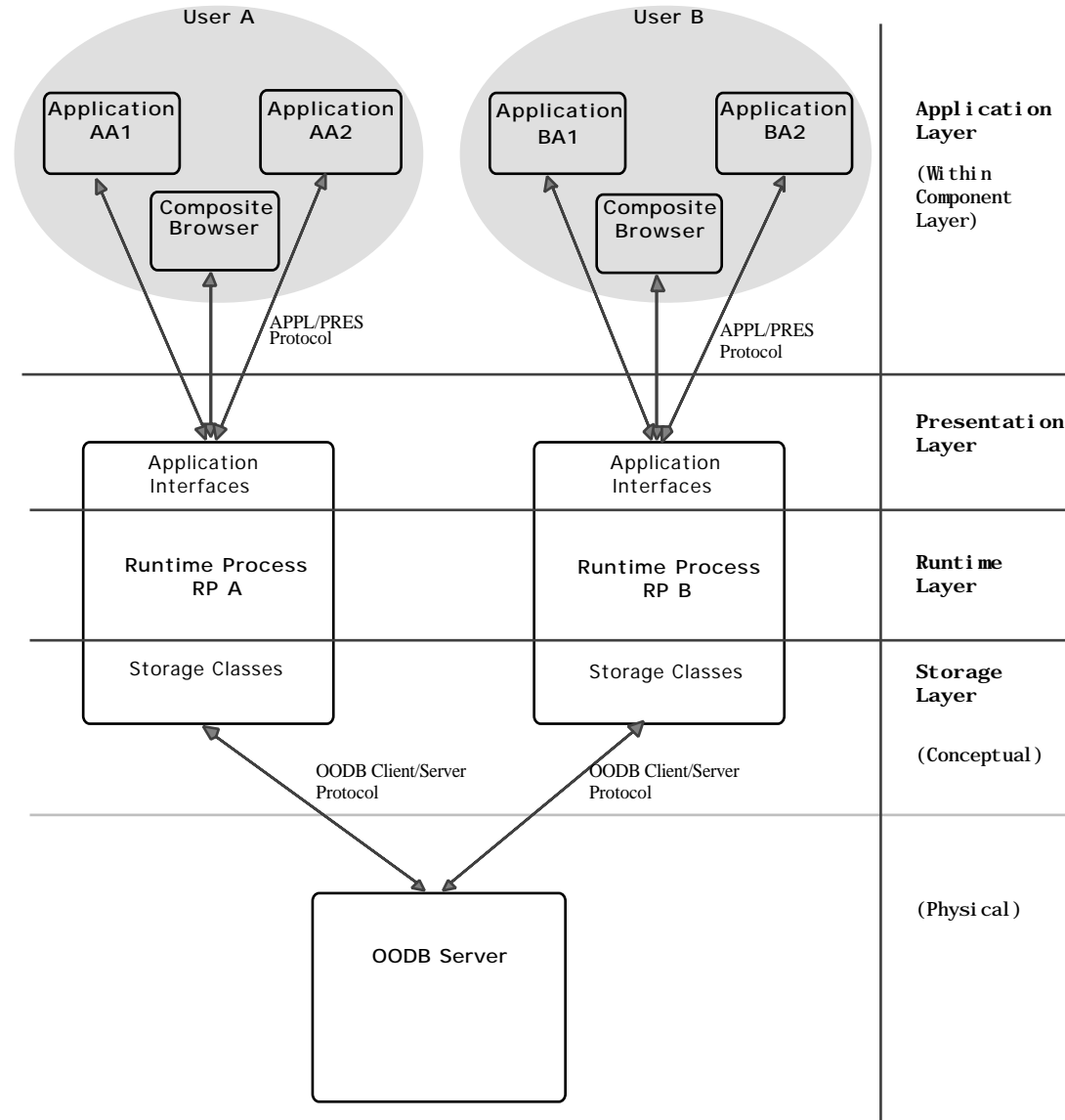
# Hypermedia in a distributed environment

---

## Naming and referencing

- **Persistent oodb object references**
  - system manages references for you, free gc
  - have to run the oodb to follow the reference
  
- **Path names (e.g. http's URL)**
  - human readable references
  - problems if files are moved, renamed
  
- **Universal name space (e.g. emerging URN standard for WWW)**
  - lets files be moved and renamed
  - requires nameservers accessible to all potential users

# Multiuser client/server architecture based on the Dexter model



# Object-Oriented Database (OODB)

---

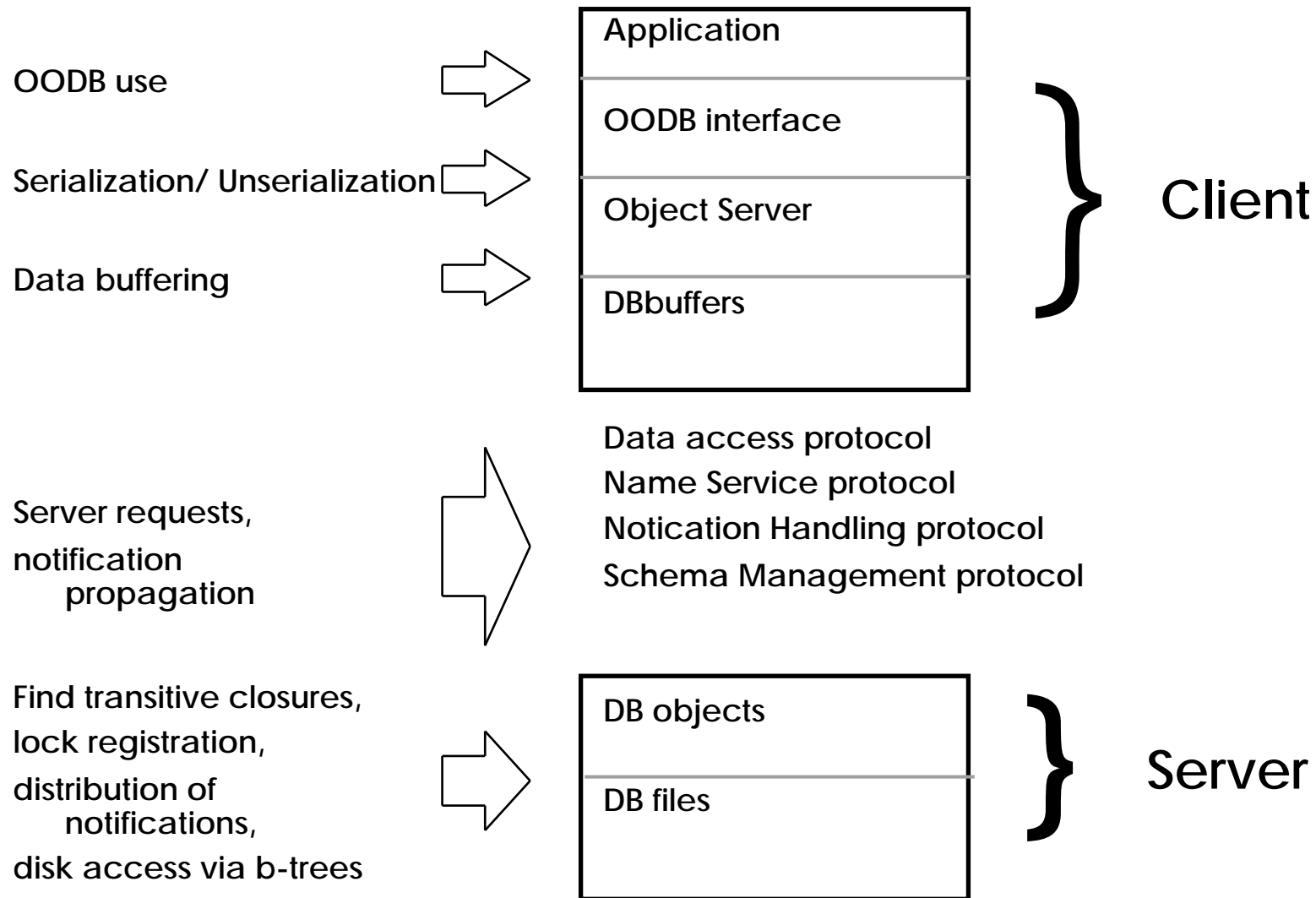
## Basic functionality:

- **General persistent store for objects specified in the Mjølner BETA language**
  - class definitions constitute the conceptual schema
- **Transparent storing and retrieval of objects including their *transitive closure***
  - Transitive closure: objects reachable through pointers

## Shared multi-user capabilities:

- **Transaction mechanism supporting multiple clients using shared servers**
  - arbitrary length transactions, parallel transactions
- **Lock management**
  - Single objects and groups of objects can be locked (write access is provided)
  - Locks can be upgraded and downgraded dynamically
- **Awareness notifications**
  - Clients can subscribe to notifications about “critical” server events
  - Notifications distributed to clients trigger appropriate reactions

# OODB Architecture Interfaces





# Approaches to persistence

---

## Our Approach:

- **Persistence transparent to application programmer**
  - Persistence applies to (almost) any object in the BETA language
  - Support for both closure and single object storing

## Alternative approaches

- **Objects created with a special 'new' operator can be persistent**
  - Examples: ObjectStore (C++), ...
- **Persistent objects inherit from common 'persistence' class**
  - Examples: GEMSTONE (SmallTalk, C), ONTOS (C++), ...
- **Objects must be inserted in special persistent container objects**
  - Examples: Prograph (Visual programming), ...

*Also different approaches to handling of OODB transactions, locking, notification, etc.*

---

# Special Topics

# Topic: Cooperative hypermedia

---

- **User access to components is managed by assignment of read/write/annotate access attributes (typically multiple readers and one writer).**
- **Changes to components (i.e. database events) can be announced to users who subscribe to notification distribution, thus supporting real-time mutual monitoring of collaborative work.**
- **Components and links can be private, owned by a group, or be public.**
- **State of component history can be inspected (who did what when?).**
- **Different users can maintain different views on shared information networks.**

# Event Notification facilities

---

**Dexter-based framework extended to support event notifications for:**

- **Creation/modification/deletion of hypertexts**
- **Creation/modification/deletion of individual components**
- **Creation/modification/deletion of anchors**
- **Lock changes for entire hypertexts and components**

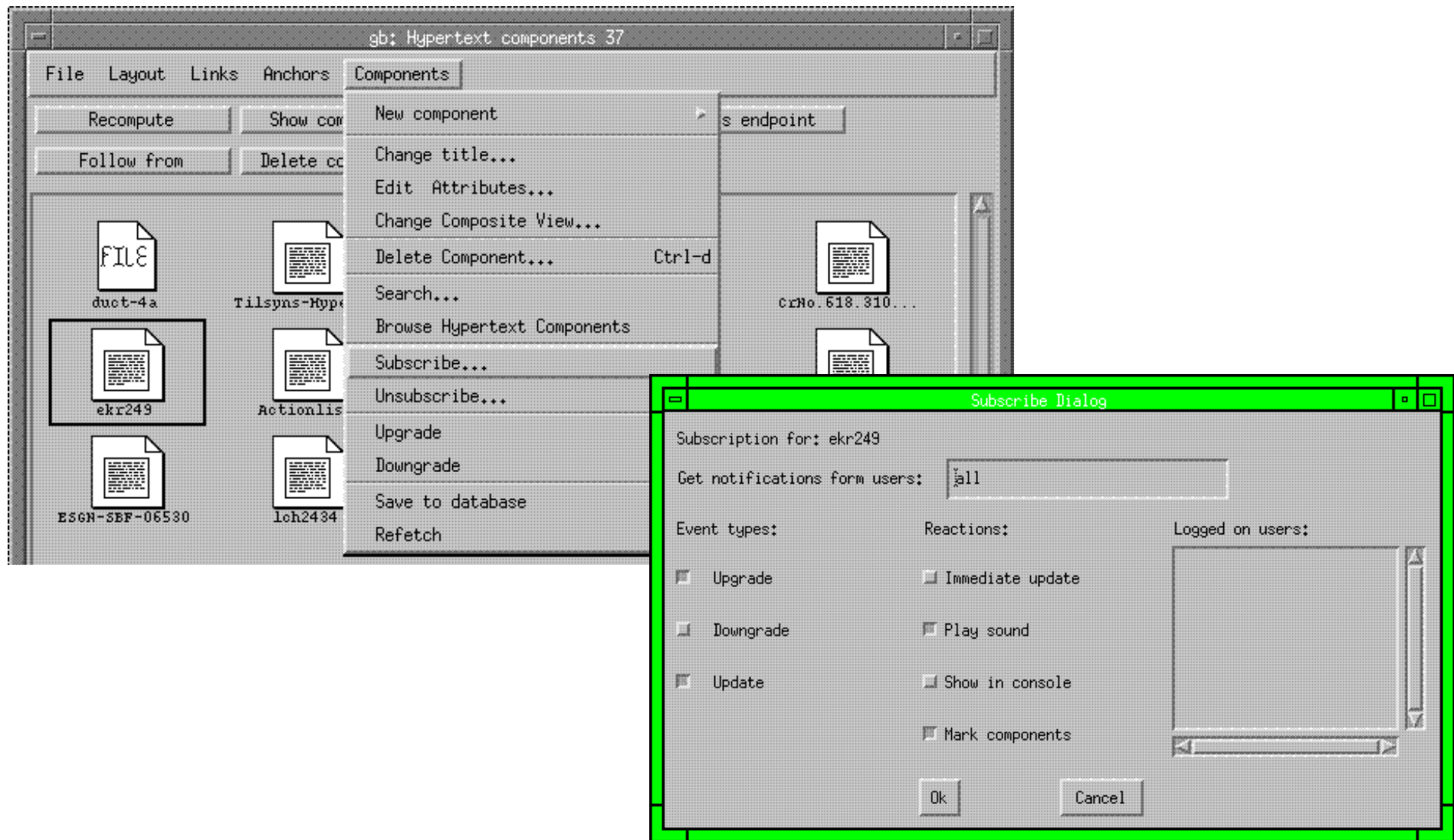
**Users subscribe to event notifications for actions by:**

- **all users**
- **a group of users**
- **an individual user**

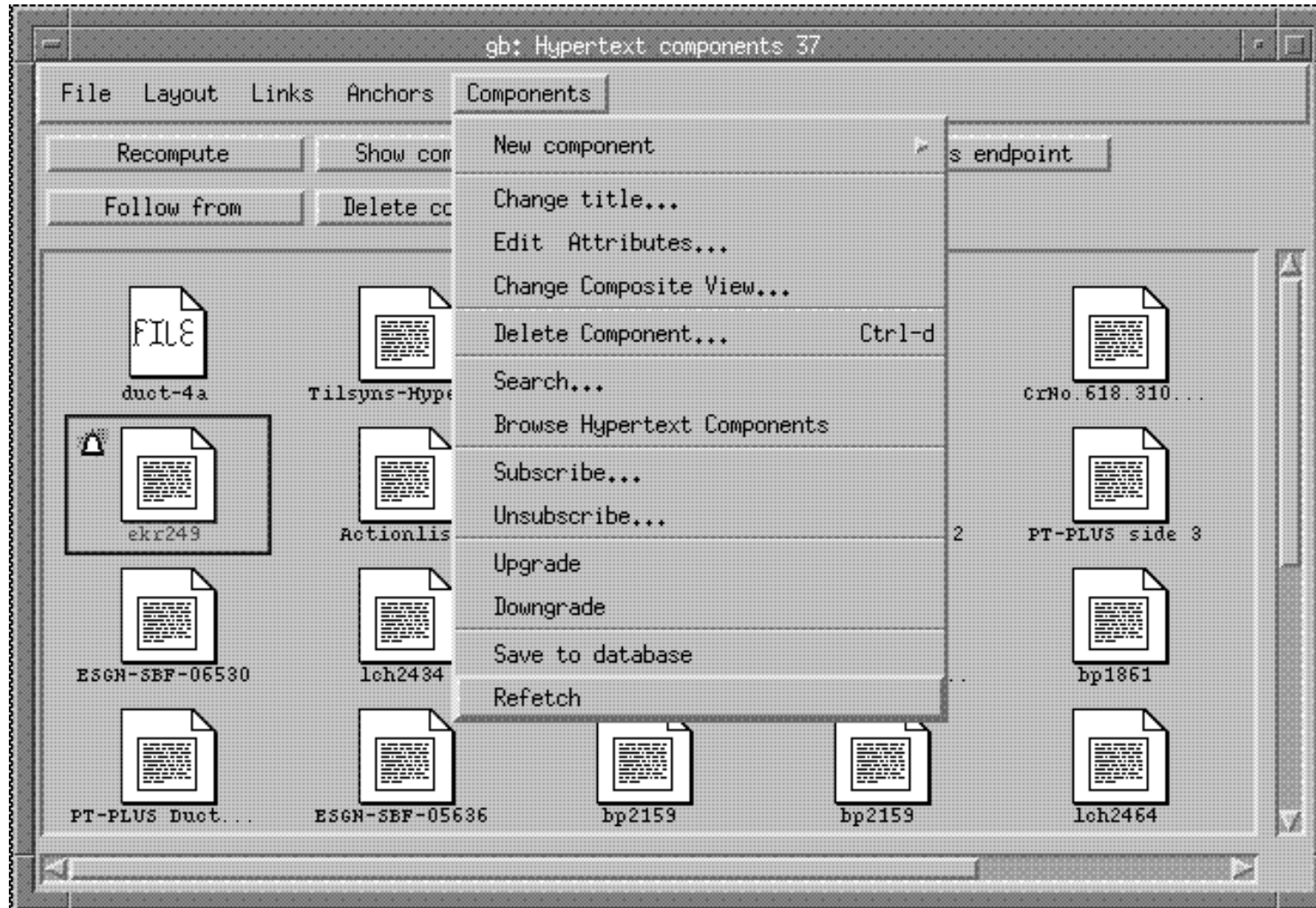
**Users choose refetch strategy:**

- **manual**
- **automatic (immediate update)**

# Subscription

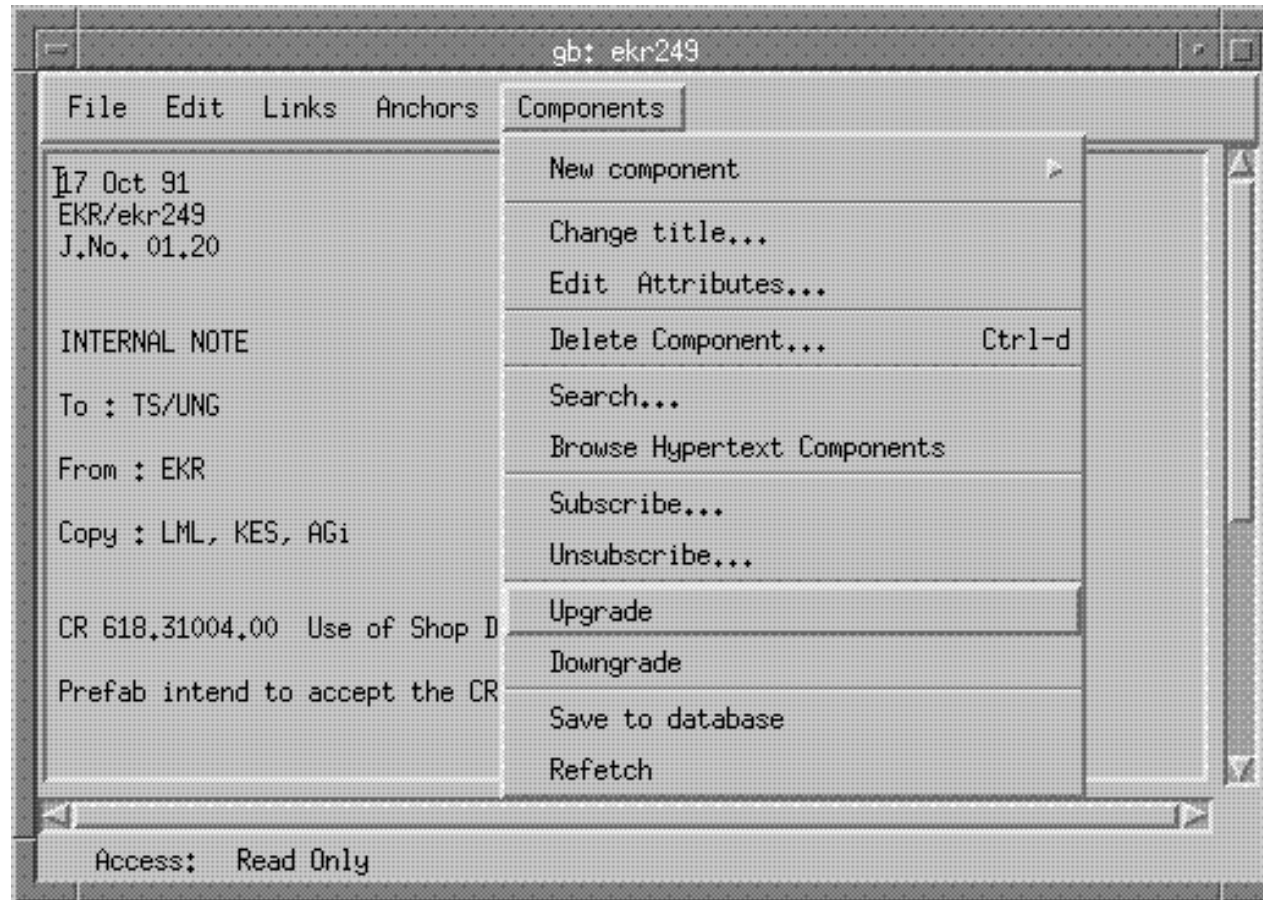


# Notification and refetch of components



# Change of lock for a component

---



# Framework extensions to handle notifications and locks

```
SessionMgr: Class
(
  subscribe:...
  unsubscribe:...
  session: Class
  (
    subscribe:...
    unsubscribe:...
    reGet:...
    changeLock:...
    instantiation: Class
    (
      subscribe:...
      unsubscribe:...
      reGet:...
      changeLock:...
      linkMarker: Class(...)
      (* instantiation private *)
      instantiationReaction: sessionReaction(...)
      updateR: instantiationReaction(...)
      ...
    )
    (* session private *)
    sessionReaction: Reaction(...)
    updateR: sessionReaction(...)
    ...
  )
  (* sessionMgr private *)
  Reaction: Class(...)
  StartR: Reaction( ...)
  ...
)
```



# Topic: Extensibility and tailorability

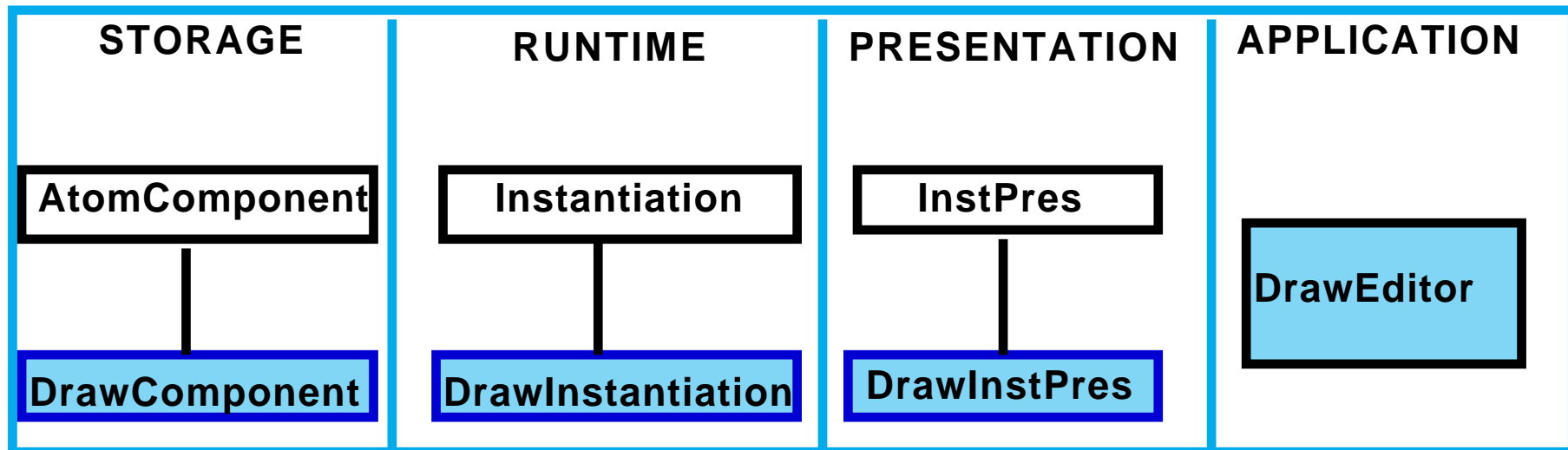
---

**An object oriented framework allow extensions and tailoring along several dimensions:**

- 1. Flexible use. The Dexter-based model can be applied to many different domains.**
- 2. "Simple" tailoring. This involves "preference sheet" choices that govern behavior of the particular UI.**
- 3. Integrating a new editor for a given instantiation.**
- 4. Building a new atomic component type. For each new component, a new instantiation type must be created.**
- 5. Building a new composite component type. This requires creation of a new instantiation type as a specialization of compositeInst.**
- 6. Building a new instantiation type.**
- 7. Building a new user interface. This is analogous to (3).**
- 8. Porting systems to a new platform.**

## Example: Adding a new media type

---



# Connecting a new editor to framework classes (1 of 2)

---

File Name : wrap-editor.eps

Title : /tmp/xfig-fig026448

Creator : fig2dev

CreationDate : Thu Jan 6 14:55:38 1994

Pages : 0

# Connecting a new editor to framework classes (2 of 2)

---

File Name : editor-details.eps  
Title : /tmp/xfig-fig008376  
Creator : fig2dev  
CreationDate : Mon Dec 27 00:40:55 1993  
Pages : 0

# Registering new classes in the framework (1 of 2)

---

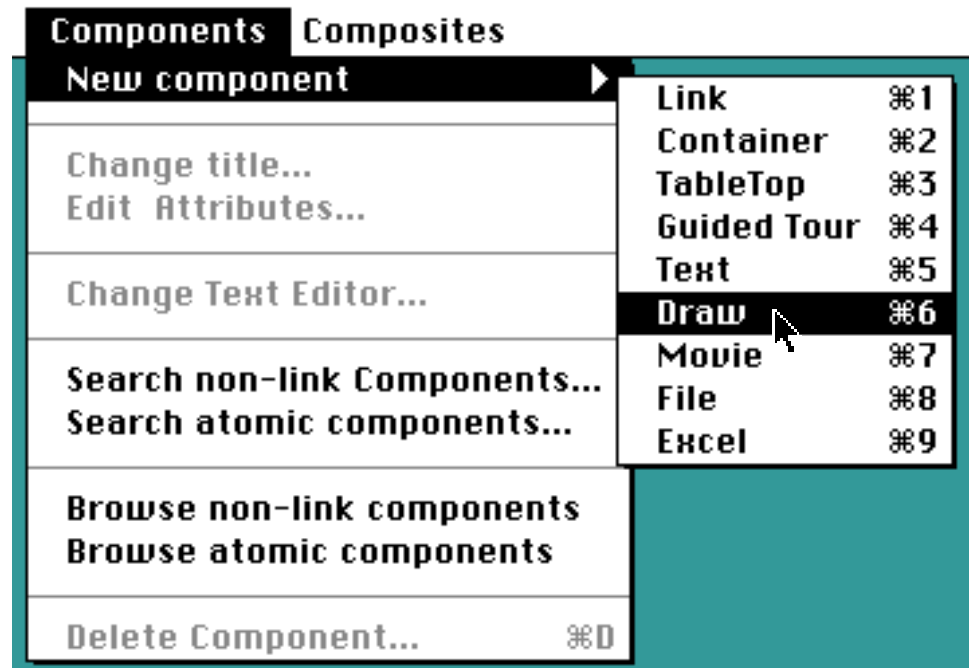
```
typeInfo: Class
(# attributes: attributeValueTable
  hasAttr: (# attr: text; found: boolean; enter attr do ... exit found #);
  getAttr: (# attr: text; val: valueType; enter attr do ... exit val #);
  setAttr: (# attr: text; val: valueType; enter (attr,val) do ... #);
  removeAttr: (# attr: text; enter attr do ... #);
  init virtual (# do ...#);
#)
...
DrawCompTypeInfo: Class AtomCompTypeInfo
(# init binding
  (# do ('Name','DrawComponent') -> setAttr;
    ('Class',DrawComponent##) -> setAttr;
    ('PrefInstTypeName','DrawInstantiation') -> setAttr;
    ...
  #);
#)
```

# Registering new classes in the framework (2 of 2)

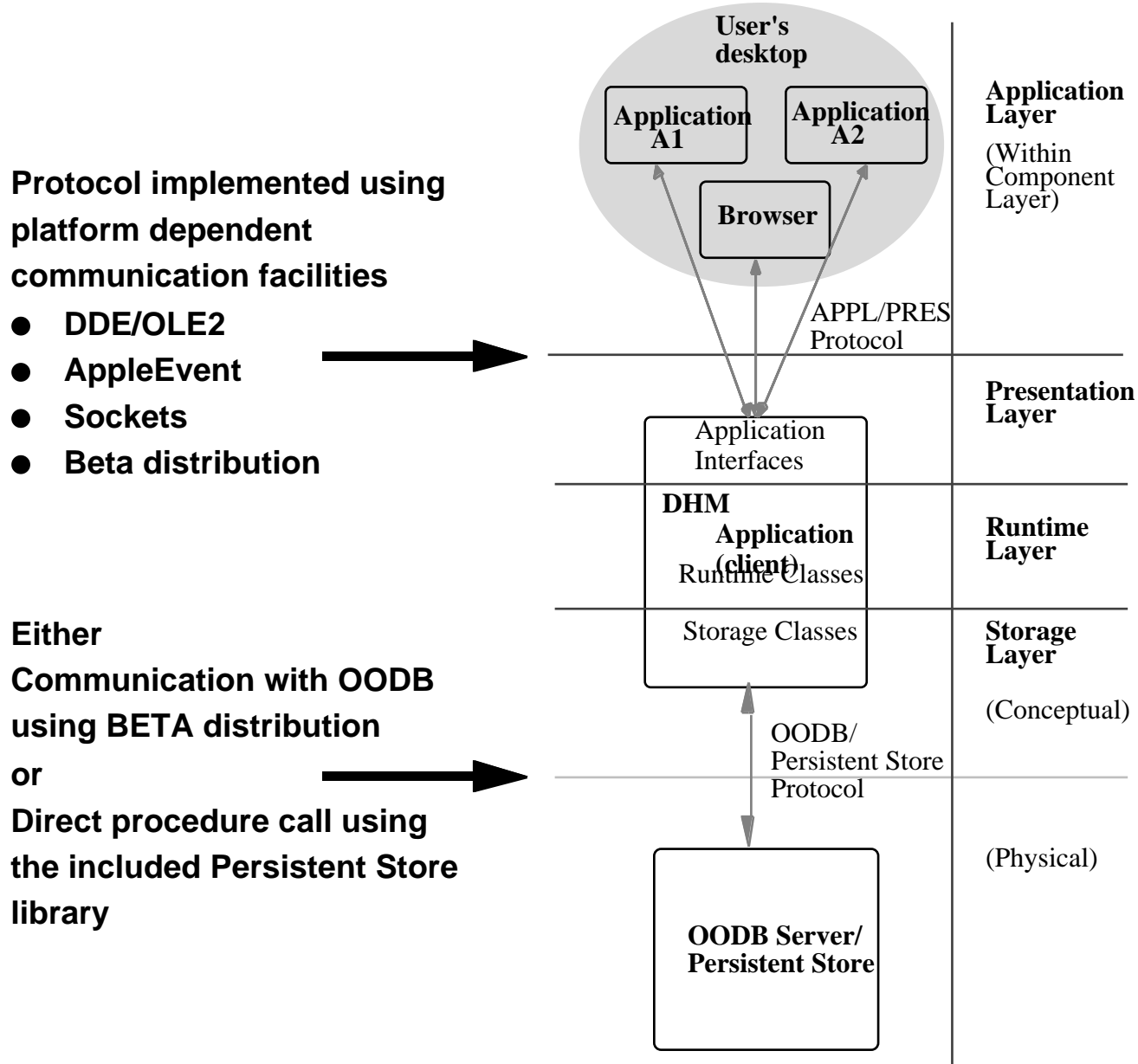
TypeInfo objects are inserted in global tables:

```
&DrawPresTypeInfo[] -> ...sessionMgrTypes.append;
...
&DrawInstTypeInfo[] -> ...sessionTypes.append;
...
&DrawCompTypeInfo[] -> ...sessionTypes.append;
...
```

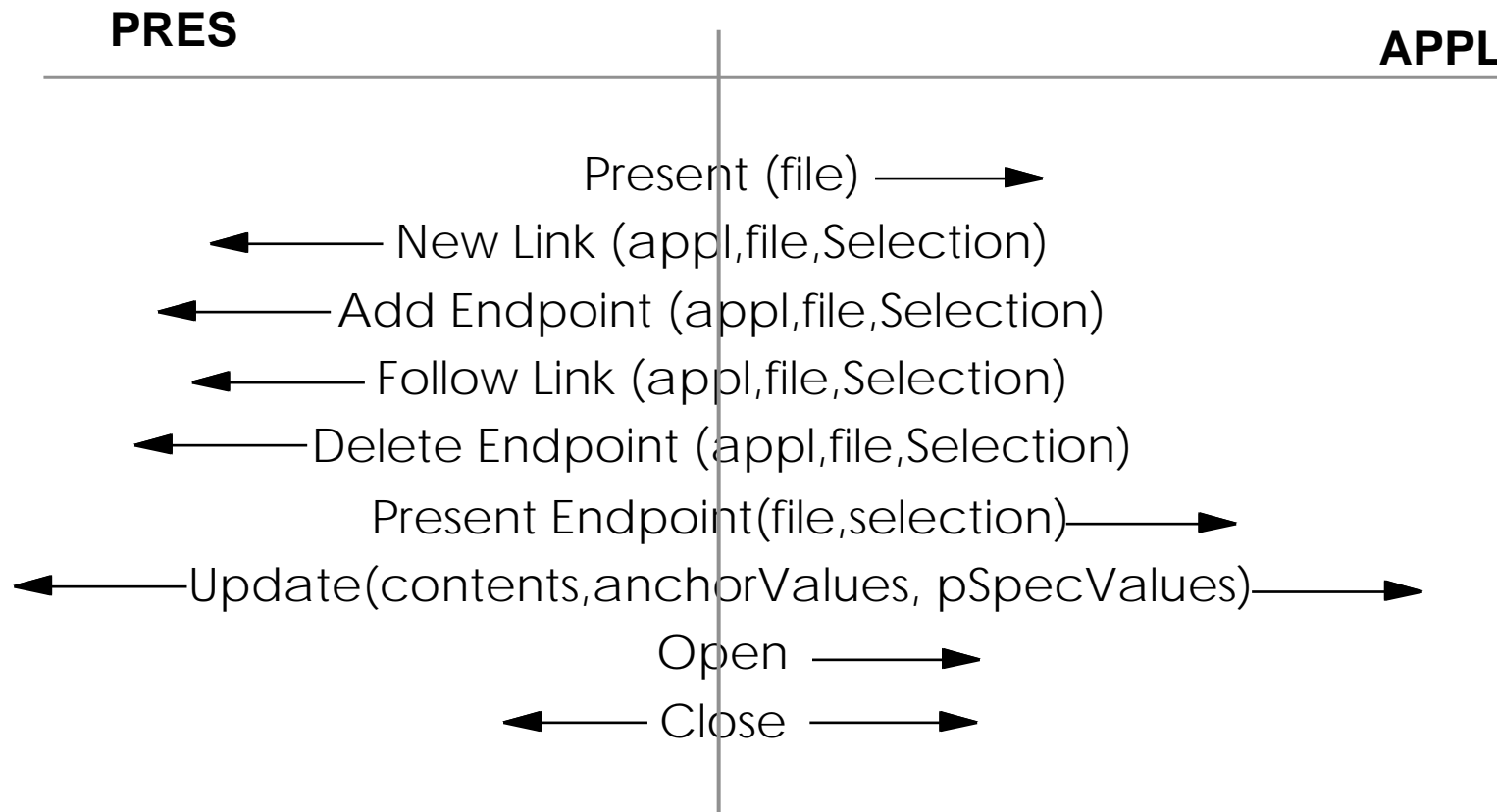
Visible result:



# Topic: Third party integration



# Simple APPL/PRES integration protocol



Example Implementation;

Excel AppleEvent with the following text data:

```
#XCL#DEVISE1:HYPERMEDIA:DEMO:TEST.XL#NL#1#R1C1:R4C5
```



# Principles for integration

---

- **Whole documents from any application may function as link endpoints (launch-only).**
- **Links to parts of documents require that the applications are open to communication via DDE, AppleEvent, etc.**
- **There should be a macro-programming language**
  - to extend the user interface
  - to communicate with the hypermedia service application
- **The applications' document format remain unmodified**
  - built in mechanisms like BookMarks, cell-names, and CAD object ID's are used as anchor values
  - positions may used as anchor values in write-protected documents.
- **The DHM client is customized for new applications using object-oriented specialization.**
- **An exception-handling mechanism is used to handle deletions of documents and of portions of documents containing links.**

# Integration experiences on several platforms

---

## ■ Windows/NT and 95

- Microsoft Word(6.0) and Excel(5.0), Bentley's Microstation (CAD)
- In progress: WordPerfect

## ■ Apple Macintosh (68K and PowerPC)

- Microsoft Excel (4.0)
- In progress: Microsoft Word(6.0) and Excel(5.0)

## ■ Sun Solaris (Unix)

- Emacs ,vi, Design/CPN, ORACLE database views, Rank Xerox's Ariel/Documentum,

Microsoft Excel - ACTIONS.XLS

File Edit View Insert Format Tools Data Hypermedia Window Help

Geneva 10 B I U

	A	B	C	D	E	F	G	H
1	V	APP CLOSED	GRP	PERS	LETTERED FROM TOC	SERIAL		
2	V	29-10-91	PI	ED	9-10-91	ESGN	SBF	560
3	V	16-11-91	PST	AGI	27-9-91	ESGN	SBF	548
4	V		PI	NCH	9-10-91	SBF	ESGN	396
5	V	16-11-91	PST	AGI	4-10-91	ESGN	SBF	556
6	V		PST	AGI	16-11-91	SBF	ESGN	4203
7	V	11-11-91	PST	AGI	26-9-91	ESGN	SBF	5482
8	V		PST	AGI	16-11-91	SBF	ESGN	4203
9	V		PI	ED	4-11-91	ESGN	SBF	5799
10	V	11-11-91	RO	AGI	24-10-91	ESGN	SBF	3726

Follow forward links from markers in current selection

Devise Hypermedia v1.5

File Hypertexts Links Components

New Link  
Add Endpoint  
Follow Link  
Delete Markers  
Inspect Markers  
Inspect Links in Workbook

Forward C+F  
Backwards C+B  
Both

gb: Component Browser

File Layout Links Anchors Components

Recompute Show component New Link from  
Follow from Delete component Browse Links

PTPCONT.DOC LYN631.DOC **PTPLUS.DOC** 056

ACTIONS.XLS  
w2223010.dgn

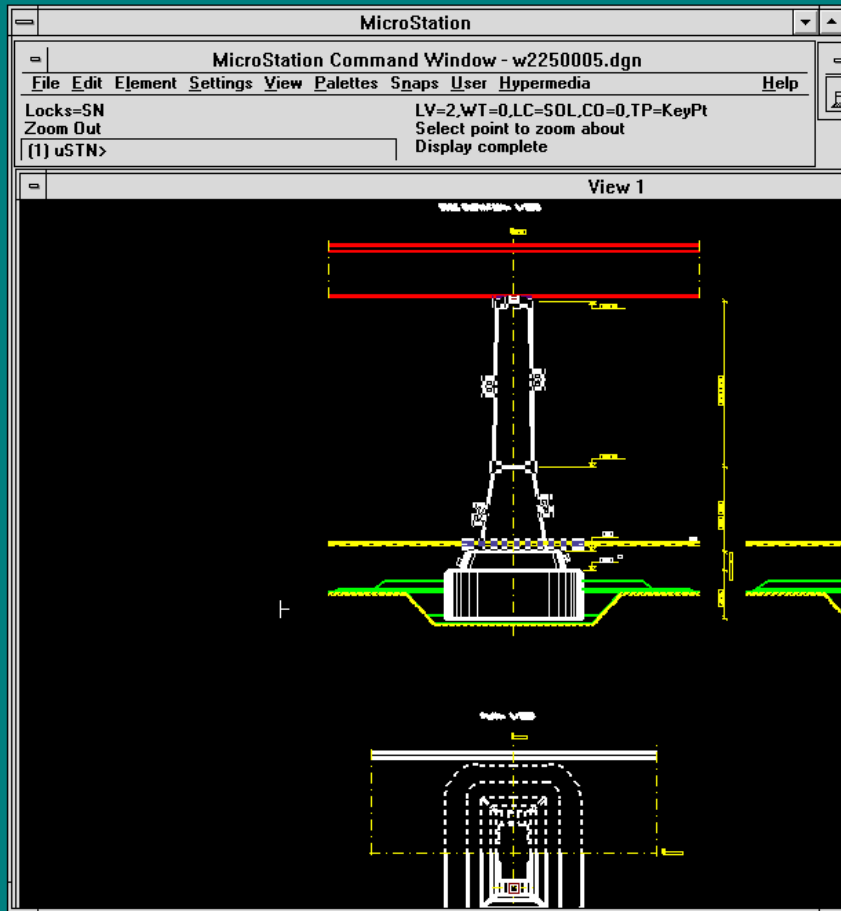
gb: Component links

File Links Anchors Components

Links for component: ACTIONS.XLS

Link 2: ( ACTIONS.XLS(S), 7384.DOC(S))  
Link 116: ( ACTIONS.XLS(S), 05636.DOC(S))

Recompute  
Edit link  
Make current  
Present link  
Delete link  
Present component



Microsoft Word - BP2159.DOC

File Edit View Insert Format Tools Table Window Hypermedia Help

Normal Times 10 B I U

Page 0

06 Aug 92  
CF/bp2159  
J.No. 01.20-18.4

**INTERNAL NOTE**

To : LML  
From: CF  
CC : KMo, UNG, CCL

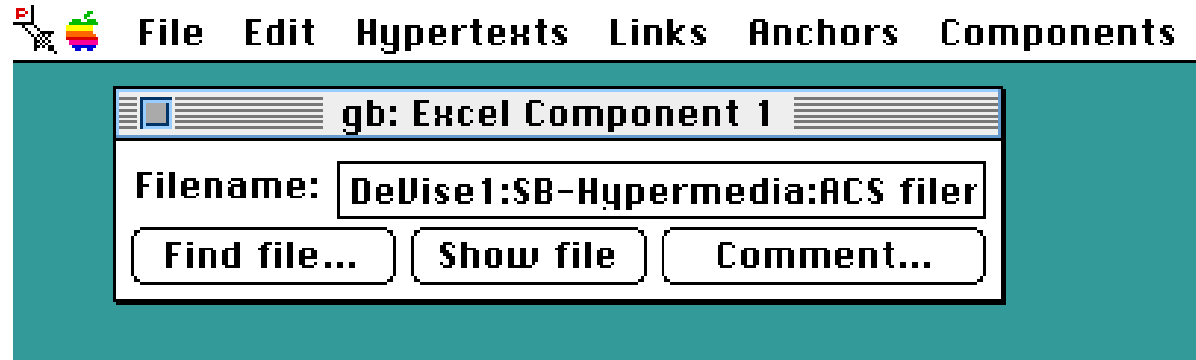
**VSL PT-PLUS Plastic Duct System**

With reference to ESG's letter ESGN-SBF-07384 we have the following comments:

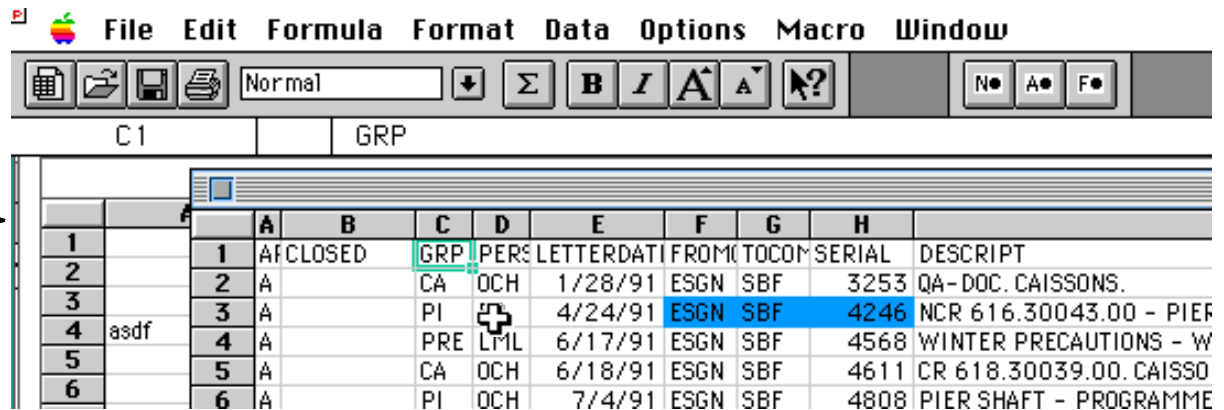
- The steel ducts used so far fulfil the specifications in the Contract except water tightness of the flattened ducts
- The proposed plastic ducts for transverse prestressing in bridge ducts for rail and roadgirders are technicalwise acceptable. They provide an improved quality regarding safety against corrosion and fatigue as well as an insignificantly higher utilization of the tendon material.

# Microsoft Excel integration

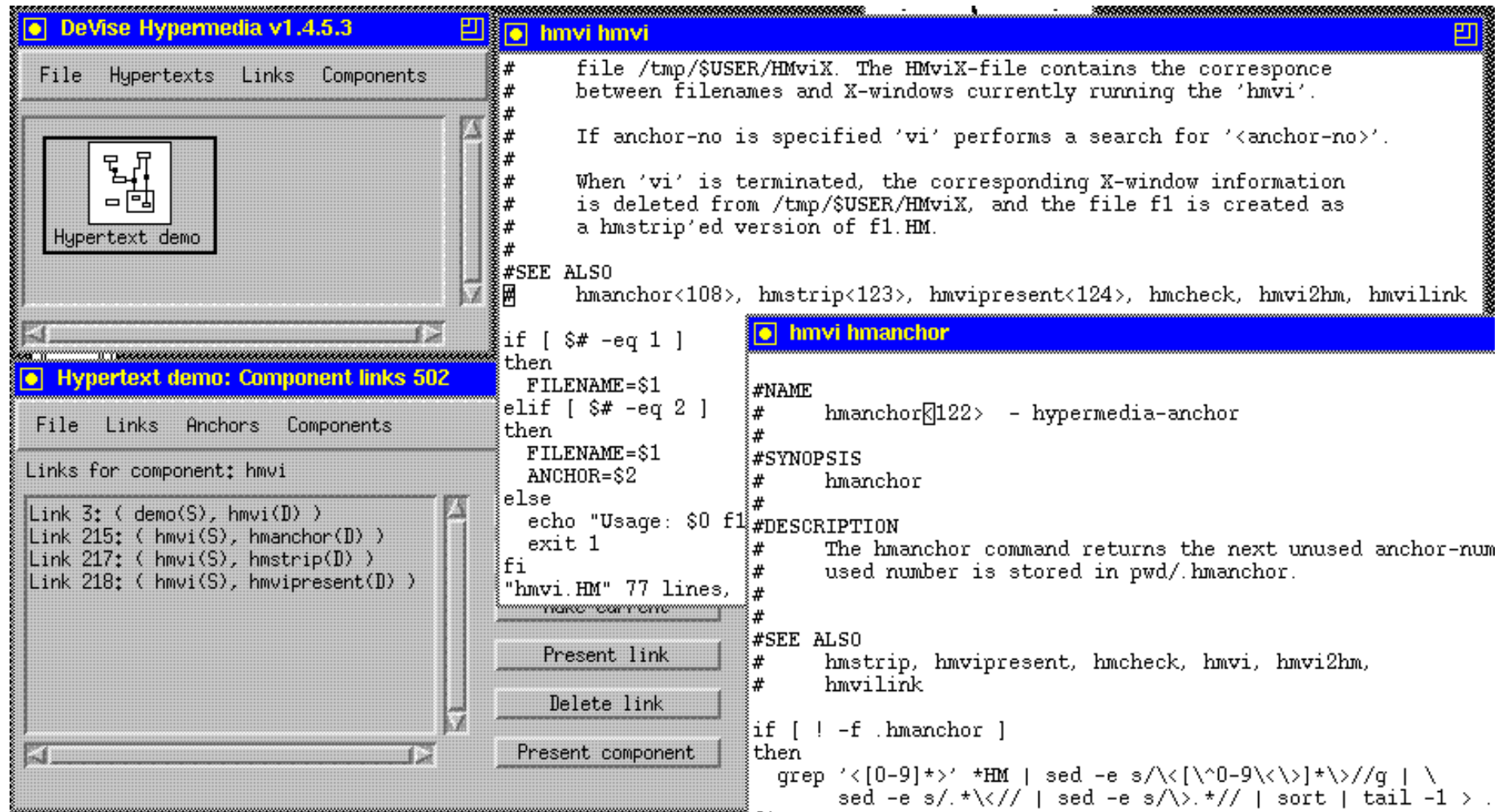
DHM Excel Component  
User Interface



Excel User Interface  
Extended with Toolbar  
buttons to communicate  
with DHM



# UNIX 'vi'-integration



The screenshot displays the DeVise Hypertext v1.4.5.3 interface. The top window, titled "DeVise Hypertext v1.4.5.3", shows a "Hypertext demo" with a diagram of interconnected nodes. Below it, a window titled "Hypertext demo: Component links 502" lists links for the component "hmvi":

- Link 3: ( demo(S), hmvi(D) )
- Link 215: ( hmvi(S), hmanchor(D) )
- Link 217: ( hmvi(S), hmstrip(D) )
- Link 218: ( hmvi(S), hmvipresent(D) )

The main editor window, titled "hmvi hmvi", contains the following shell script:

```
# file /tmp/$USER/HMviX. The HMviX-file contains the correspondence
# between filenames and X-windows currently running the 'hmvi'.
#
# If anchor-no is specified 'vi' performs a search for '<anchor-no>'.
#
# When 'vi' is terminated, the corresponding X-window information
# is deleted from /tmp/$USER/HMviX, and the file fl is created as
# a hmstrip'ed version of fl.HM.
#
#SEE ALSO
# hmanchor<108>, hmstrip<123>, hmvipresent<124>, hmcheck, hmvi2hm, hmvilink

if [ $# -eq 1 ]
then
  FILENAME=$1
elif [ $# -eq 2 ]
then
  FILENAME=$1
  ANCHOR=$2
else
  echo "Usage: $0 fl"
  exit 1
fi
"hmvi.HM" 77 lines,
make current

Present link
Delete link
Present component
```

The "hmvi hmanchor" window shows the following shell script:

```
#NAME
# hmanchor[122] - hypermedia-anchor
#
#SYNOPSIS
# hmanchor
#
#DESCRIPTION
# The hmanchor command returns the next unused anchor-num
# used number is stored in pwd/.hmanchor.
#
#SEE ALSO
# hmstrip, hmvipresent, hmcheck, hmvi, hmvi2hm,
# hmvilink

if [ ! -f .hmanchor ]
then
  grep '<[0-9]*>' *HM | sed -e s/\([\^0-9\<>]*\)//g | \
  sed -e s/.*\<\/ | sed -e s/\>.*\/ | sort | tail -1 > .
```

# Further issues on third party integration

---

**We have taken significant steps towards providing an extensible Dexter-based hypermedia service to third party applications**

- basic anchor based linking can be provided
- experiences from several platforms and application types

**But there is still a lot of open issues**

- **How to make the linking service more application independent?**
  - standardized use of anchor and PSpec values
- **How to avoid or detect invalid anchor values in third party applications?**
- **How to combine the multi-user facilities of the hypermedia service with filesystem access control and sharing?**
- **How to support cross platform hypermedia support for external files and third-party applications?**
- **How do we efficiently integrate with World Wide Web documents?**

---

# Wrap Up

# Wrap-up

---

- In today's "brave new world" where integration, tailorability and open implementations are the operative buzzwords, we believe the Dexter model can make significant contributions.
- In particular, Dexter can provide a starting point for hypermedia design, offering a framework for thinking about concepts like anchors, composites, & integration.
- Dexter has problems (and gaps) which need to be addressed by developers of object-oriented hypermedia systems including, for example, CSCW issues.
- An example of a Dexter-based system is DHM: see contact information at the end...



# Research topics

---

## ■ Browsing:

- How to display complex structures built from multi-headed links?

## ■ Versioning:

- How to handle versions of (parts of) hypertexts referencing lots of external files managed by a conventional filesystem?

## ■ Queries:

- How to combine structural search performed by hypermedia engine with contents search performed by third party applications?

## ■ Integration of third party applications:

- Multiuser, cross platform, invalid anchor values, WWW ...

## ■ Multimedia:

- How to integrate with advanced multimedia design environments?

## ■ Distribution:

- How to support smooth transitions from global distributed hypermedia (e.g. Internet) to local hypermedia systems embedded in particular organizations?

# Monolithic versus open systems (Halasz, 1990)

---

File Name : monolithic75.eps

Title : /tmp\_mnt/tilde/trigg/Dexter/ECHT-tutorial/nist-slides/monolithic.ps

Creator : trigg

CreationDate : Wed Aug 31 01:17:57 1994

Pages : 1

# State of *DEVISE* Hypermedia (DHM)

---

*Single user version used in workshops at Great Belt Link Ltd. (and Grundfos)*

**DHM for Macintosh**

*Network version evaluated at GBL spring 95*

**DHM for Windows/NT/95**

*Multi user version in use by EuroCODE partners*

**DHM for UNIX**

*Multi user prototype - demo at Hypertext '93*

**DHM for Mac/UNIX**

---

**Portable Dexter based hypermedia framework**

---

**Single user Persistent store**

**Multi-user OODB**

*Development is based on the Scandinavian Mjølner BETA environment*

---

# How to obtain further material and developer license for the DHM Framework etc.

---

Requests for information about the DHM Framework should be addressed to:

**Kaj Grønbæk,  
Computer Science Department,  
Aarhus University,  
Ny Munkegade , Bldg. 540,  
DK-8000 Århus C  
Phone: +45 8942 3237  
Fax: +45 8942 3255  
E-mail: [kgronbak@daimi.aau.dk](mailto:kgronbak@daimi.aau.dk)  
WWW: <http://www.daimi.aau.dk/~kgronbak>**

Requests for information about the Mjølner BETA system should be addressed to:

**Mjølner Informatics Aps.  
Science Park  
Gustav Wieds Vej 10,  
DK-8000 Århus C  
Phone: +45 8620 2020  
Fax: +45 8620 1222  
E-mail: [info@mjolner.dk](mailto:info@mjolner.dk)**

# A few references

---

- Grønbæk, K. and Malhotra, J. Building Tailorable Hypermedia Systems: The embedded-interpreter approach. To appear in proceedings of the ACM conference on Object Oriented Programming Systems, Languages and Applications (OOPSLA '94). Portland, Oregon, 23-27 October, 1994.
- Grønbæk, K. and Mogensen, P. Specific cooperative analysis and design in general hypermedia development. To appear in Proceedings of the Participatory Design Conference (PDC '94), Chapel Hill, North Carolina, USA, October 27-28, 1994. Computer Professionals for Social Responsibility.
- Grønbæk, K. and Trigg, R.H. Design issues for a Dexter-based hypermedia system. Communications of the ACM 37, 2 (Feb. 1994), pp. 40-49.
- Grønbæk, K. and Trigg, R.H. Hypermedia System Design Applying the Dexter Model Guest editors' introduction to the special issue on Hypermedia in Communications of the ACM 37, 2 (Feb. 1994), pp. 26-29.
- Grønbæk, K. Composites in a Dexter-Based Hypermedia Framework. To appear in proceedings of the ACM European Conference on Hypermedia Technology (ECHT '94) Edinburg, UK, September 18-23, 1994.
- Grønbæk, K., Hem, J.A., Madsen, O.L., and Sloth, L. Cooperative Hypermedia Systems: A Dexter-based architecture. Communications of the ACM 37, 2 (Feb. 1994), pp. 64-75.
- Grønbæk, K., M. Kyng, and P. Mogensen, CSCW Challenges: Cooperative Design in Engineering Projects. Communications of the ACM, 1993. 36(6): p. 67-77.
- Grønbæk, K. and Trigg, R.H. Toward a Dexter-based model for open hypermedia: Unifying embedded references and link objects. In proceedings of HYPERTEXT '96 – Seventh ACM Conference on Hypertext, Washington DC, USA, March 16-20, 1996.
- Halasz, F. (1990). The Dexter Hypertext Reference Model, Presentation at The Hypertext Standardization Workshop. Gaithersburg, MD: National Institute of Standards.
- Halasz, F. and Schwartz, M. (edited by Grønbæk, K. and Trigg, R.H.) The Dexter Hypertext Reference Model. Communications of the ACM 37, 2 (Feb. 1994), pp. 31-39.
- Halasz, F., & Schwartz, M. (1990). The Dexter Hypertext Reference Model. In J. Moline, D. Benigni, & J. Baronas (Eds.), Proceedings of The Hypertext Standardization Workshop (pp. 95-133). Gaithersburg, MD: National Institute of Standards. January.
- Hardman, L., Bulterman, D.C.A., and van Rossum, G., The Amsterdam Hypermedia Model: Adding Time and Context to the Dexter Model. Communications of the ACM, 1994. 37(2): p. 50-63.
- Leggett, J. and Schnase, J. Viewing Dexter with Open Eyes. Communications of the ACM 37, 2 (Feb. 1994), pp. 77-86.
- Newcomb, S. R., Kipp, N. A., & Newcomb, V. T. (1991). The "HyTime" Hypermedia/Time-based Document Structuring Language. Communications of the ACM, 34(11), 67-83. November.
-