Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

Hypermedia
Future Concepts

- Starting Point: The Web
- Markup, XML and related Technologies
- Open Hypermedia Architectures
- Linking and Anchoring
- Authoring
- Application Example

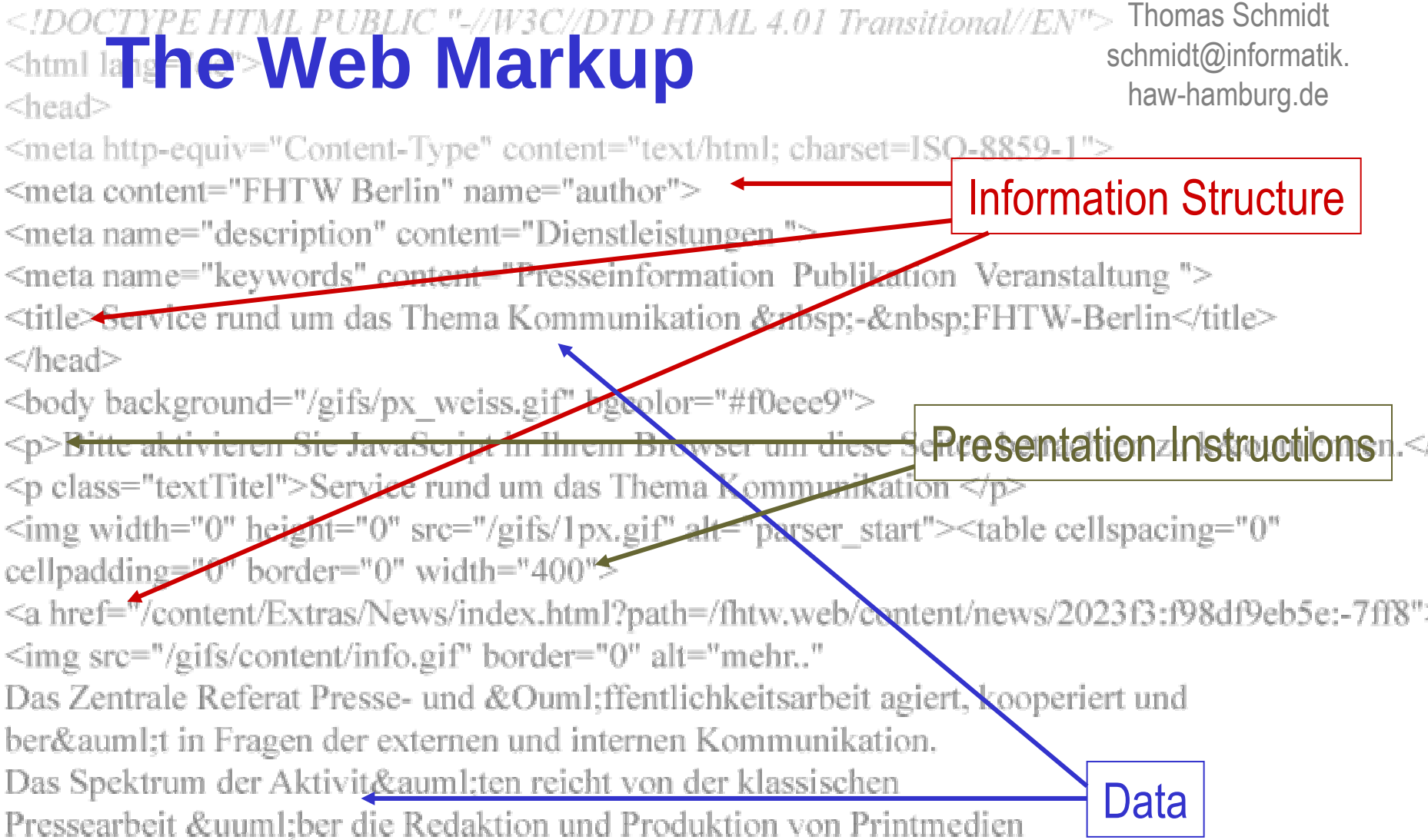
Presence: The Web

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- HTTP – fairly general, as specific protocols can be embedded
- URI - fairly useful, but limited # of namespaces available
- Markup - html ... the Problem

The Web Markup

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de



Can WWW be succesful?

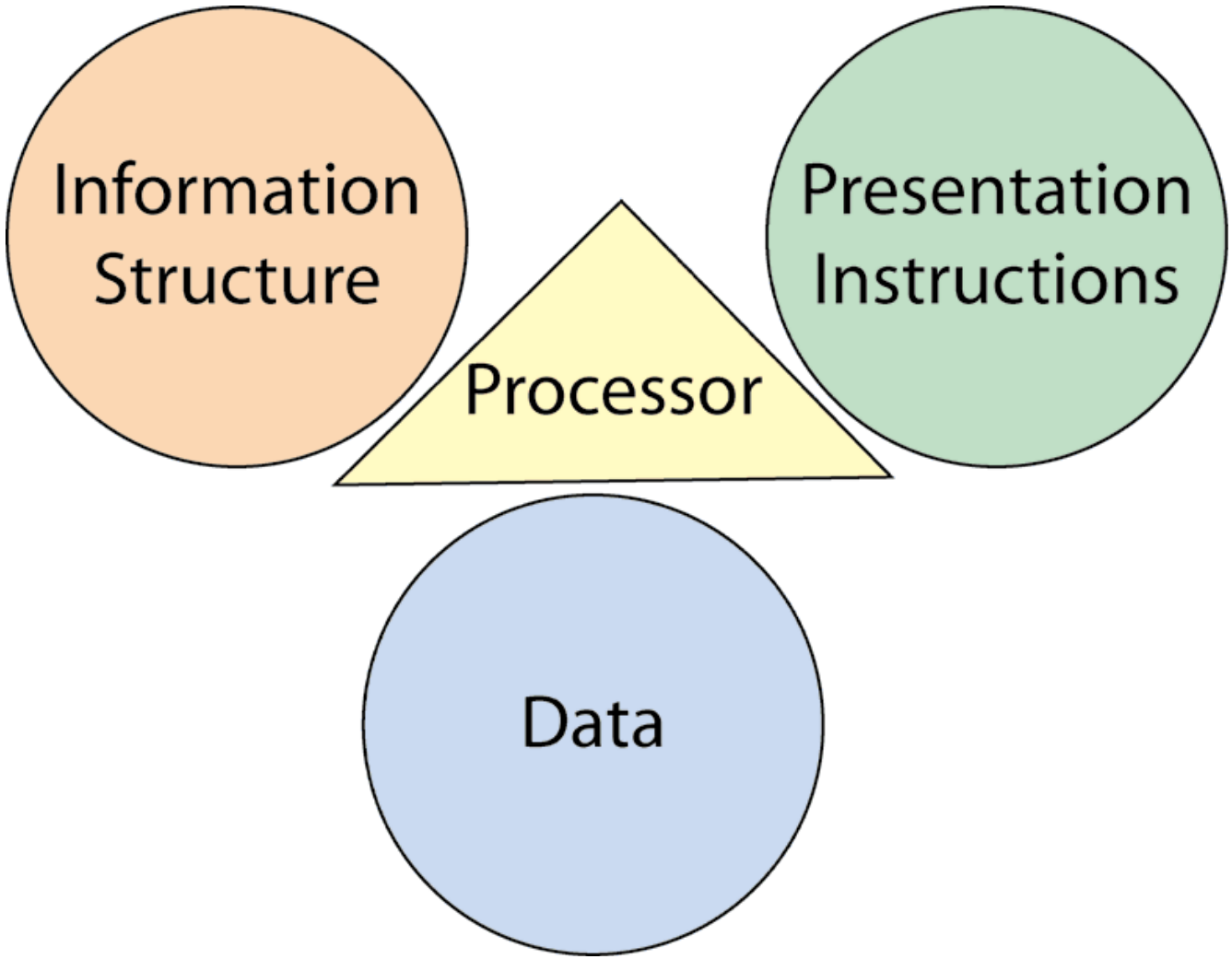
Herrmann Maurer 1999

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- The Web, i.e. servers & browsers, are everywhere
- HTML is a de facto standard
- The Web brought up a new standard trail for hypermedia: The W3C consortium: www.w3.org (1994)
- This standardisation consortium probably was the most relevant outcome for hypermedia technology
  (even though it is intricate to get involved in W3C)

XML

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de



What is XML?

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- eXtensible Markup Language
- W3C standard: www.w3.org/XML
- Metaconcept to define markup languages
- Purpose:
  - simpleness of HTML
  - functionality of SGML
- Structuring by using a DTD or an XML Schema

XML Validation

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- Well-formed
  - A document that conforms to the XML syntax rules
    - All XML documents must have a root element
    - All elements must be properly nested within each other
    - All elements must have a closing tag
    - attribute values must always be quoted ...
- Valid
  - Validated against a DTD or XSD is valid XML
  - Conforms to the rules of a Document Type Definition (DTD) or XSD (XML Schema Definition)

Document Type Definition

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

```
<?xml version="1.0"  encoding="UTF-8" ?>
<!ENTITY % formation "#PCDATA | bold" >
<!ELEMENT bold (#PCDATA) >
<!ELEMENT document (paragraph)+ >
<!ELEMENT paragraph ((heading)?,(text)+) >
<!ATTLIST paragraph
    align #CDATA #REQUIRED >
<!ELEMENT heading (#PCDATA) >
<!ELEMENT text (%formation;)* >
```

DTD

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE document SYSTEM "document.dtd">
<document>
    <paragraph align="left">
<heading>Überschrift</heading>
            <text>
                Der Text
<bold>fette</bold>des Paragraphes. Und so
weiter...
            </text>
    </paragraph>
</document>
```

XML document

Structuring: XML Schema

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- XML based model description language as a successor of Document Type Definition (DTD)
- Formally describes an XML grammar
- XML Schema are defined in XML Schema Definition (XSD) a W3C Recommendation on 2 May 2001
  http://www.w3.org/TR/xmlschema-0/
- XSD-Namespace:
  http://www.w3.org/2001/XMLSchema

Introduction to XSD

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- XML Schema describes design of XML documents (like DTDs)
  - Permitted elements and element nesting
  - Element occurrence constraints
  - Permitted attribute
  - Attribute types and default values
- In addition to DTDs XSD schema provides
  - Simple and complex data types
  - Declaration of new data types
  - Type derivation and inheritance
  - Namespace-aware element and attribute declaration
- Generally used in XML application frameworks (s.a. Web Services)

Usage and General Design

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- Association of XML Schema to an XML document

```
<?xml version=„1.0“?>
<doc   xmlns=„http://myUrl.org“
       xmlns:xsi=„http://www.w3.org/2001/XMLSchema-instance“
       xsi:schemaLocation=„http://myURL.org doc.xsd“>
</doc>
```

- The <schema> element is always the root of a XML Schema, containing declarations of used namespaces

```
<xs:schema xmlns:xs=„http://www.w3.org/2001/XMLSchema“
           targetNamespace=„http://myURL.org“
           xmlns=„http://myURL.org“
           elementFormDefault=„qualified“>
...
</xs:schema>
```

Why XSD Schema?

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- XSD supports data types
  - Specify content formats
  - Validate, restrict and convert content
  - Collaborate with databases
- XSD uses XML syntax
  - Appropriate self-consistent format
  - To be handled with common XML editors
  - Easily processable with common XML technologies
- XSD is extensible
  - Reuse in other XML schema
  - Build own data types on top of default types

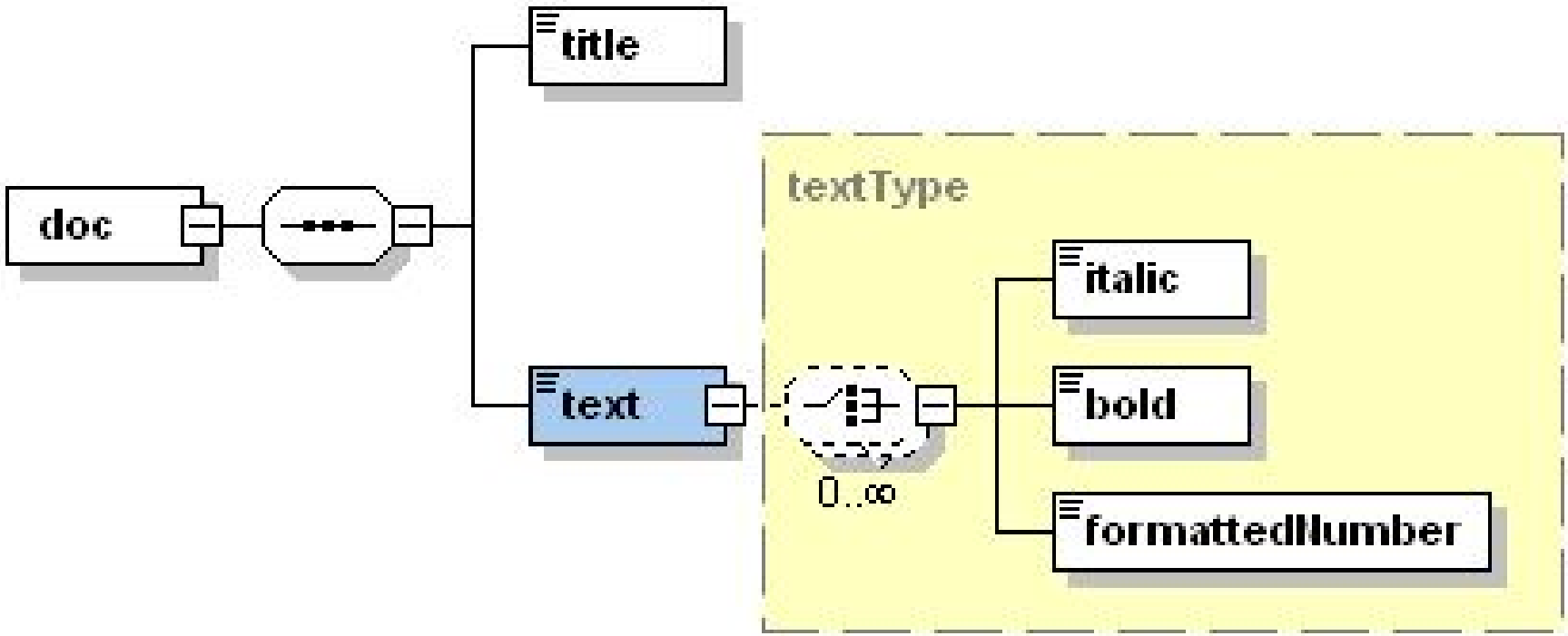
Simple Example

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

```
<xs:schema elementFormDefault="qualified"
   xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name=„doc“>
   <xs:complexType>
      <xs:element name=„title“ type=„xs:string“/>
      <xs:element name=„text“ type=„textType“/>
   </xs:complexType>
</xs:element>

<xs:complexType name=„textType“ mixed=„true“/>
  <xs:choice minOccurs=„0“ maxOccurs=„unbounded“>
    <xs:element name=„italic“ type=„xs:string“/>
    <xs:element name=„bold“ type=„xs:string/>
    <xs:element name=„formattedNumber“ type=„xs:decimal“/>
  </xs:choice>
</xs:complexType>

</xs:schema>
```

Visual Example

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de



Elements & Types

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- Types are like classes in object-oriented data models
- Named types can be reused in multiple locations as they may be referenced by elements or attributes
- Elements are named instances of user-defined or build-in types (similar to objects in object-oriented models)
- Element and type declarations can be locally or globally available
- Globally defined elements can be referenced at appropriate location

Simple & Complex Types

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- XML Schema distinguishes between simple and complex types / elements
- Simple types contain neither attributes nor child elements
- Simple types represent all kinds of data types as text
- Complex types may be built from all possible combinations of simple types, attributes and child elements
- Attributes must always be declared as simple types

Simple Types

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- Simple values as string representation, where derivation and restrictions are based on build-in data types

XML Schema Definition:

```
<xs:element name=„aggreationLevel“ type=„xs:nonNegativeInteger“/>
<xslelement name=„date“ type=„xs:dateTime“/>

<xs:simpleType name=“String">
  <xs:restriction base="xs:string">
    <xs:maxLength value="255"/>
  </xs:restriction>
</xs:simpleType>
<xs:element name=“title” type=“String”/>
```

XML Sample:

```
<aggregationLevel>10</aggregationLevel>
<date>2004-09-07</date>
<title>The Beauty of 000…000</title>
```

Complex Types

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- XML Schema types, that contain other elements and / or attributes
- Element content types can be separated in four groups:
  - Empty elements
  - Elements that contain only sub elements
  - Elements that contain text and attributes
  - Mixed elements with text, sub elements and attributes

Empty Elements

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

Defined as complex types without child elements

XML Schema Definition:

```
<xs:element name=„product“ type =„productType“/ >
<xs:complexType name=„productType“  mixed=„false“/>
   <xs:attribute name=„prod_id“ type=„xs:positiveInteger“/>
</xs:complexType>

<xs:element name="br" minOccurs="0">
  <xs:complexType mixed="false">
    <xs:complexContent/>
  </xs:complexType>
</xs:element>
```

XML Sample:

```
<product  prod_id=„11111“/>
<br/>
```

Elements with Children

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

Defined as complex type that contains child elements as complex content

XML Schema Definition:

```
<xs:element name=„doc“>
  <xs:complexType>
    <xs:complexContent>
      <xs:sequence>
        <xs:element name=„title" type=„xs:string"/>
        <xs:element name=„text" type=„textType"/>
      </xs:sequence>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

XML Sample:

```
<doc>
  <title>XML Sample</title>
  <text>Hello World</text>
</doc>
```

Elements with Text and Attributes

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- Declared as complex type with a simple content part (based on simple types) that contains attributes
- Attributes are handled like simple elements, used only in complexTypes

XML Schema Definition:

```
<xs:element name=„text“>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base=„xs:string“>
        <xs:attribute name=„ID" type=„xs:integer"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

XML Sample:

```
<text ID=„654321“>Hello World</text>
```

Mixed Elements

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

<mixed> Attribute of complexType indicate concurrent use of elements, attributes and text

XML Schema Definition:

```
<xs:element name=„text“>
  <xs:complexType mixed=„true“>
    <xs:choice maxOccurs=„unbounded“>
      <xs:element name="italic" minOccurs="0"/>
      <xs:element name="bold" minOccurs="0"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

XML Sample:

```
<text>
   XML <italic>this</italic> <bold>came</bold>
   <italic>true</italic>.
</text>
```

Other XSD Components

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- Annotations
  - The <annotation> element permits human-readable (<documentation>) and / or machine-readable (<appInfo>) annotations
- <any> Element
  - Denote an element wildcard in a model group, used for open content models
- Element substitution
  - The attribute <substitutionGroup> in the head element declaration allows to define other valid element names
  - Acts like aliasing
  - Used under internationalization aspects

Samples

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

Annotations

```
<xs:annotation>
  <xs:documentation>Title of Document</xs:documentation>
  <xs:appInfo>attrClass=Default.docTitle</xs:appInfo>
</xs:annotation>
```

Any element

```
<xs:complexType name=“doc">
   <xs:sequence>
        <xs:element name=„text" type:"xs:string"/>
        <xs:any minOccurs=„0“/>
   </xs:sequence>
</xs:complexType>
```

Element substitution

```
<xs:element name=„title“ type=„xs:string“/>
<xs:element name=„Titel“ substitutionGroup=„title“>

<xs:element name=„title“ type=„xs:string“
   block=„substitution“>
```

XSD Indicators

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- Indicators permits structural control of element placement within XML documents
- Order indicators:
  - <choice> One of the specified elements
  - <all> All elements in arbitrary order
  - <sequence> All elements in specified order
- Occurrence indicators
  - <maxOccurs> / <minOccurs>
    Minimum and maximum number of times the surrounding construct can appear
- Group indicators
  - <group> / <attributeGroup>
    Combines elements as well as attributes to groups

Indicator Examples

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

```
<xs:all>
   <xs:element name=“firstname" type:"xs:string"/>
   <xs:element name=“lastname" type:"xs:string"/>
</xs:all>

<xs:group name=„docGroup“>
<xs:choice minOccurs=„1“ maxOccurs=„unbounded“>
   <xs:element name=„title“ type=„xs:string“/>
   <xs:element name=„text“ type=„xs:string“/>
</xs:choice>
</xs:group>

<xs:element name=„doc“>
  <xs:sequence>
    <xs:group ref=„docGroup“/>
    <xs:element name=„test“/>
  </xs:sequence>
</xs:element>
```

Facets / Restrictions

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- Facets are aspects of possible values for simple types
- Facets are mainly expressed as restrictions to control acceptable content for XML elements
- Limitations can be applied to:
  - Values (upper and lower boundaries)
  - Enumerations
  - Restrictions inside of content (patterns)
  - White space handling (preserve/replace/collapse)
  - Size and length restrictions

Restriction Samples

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

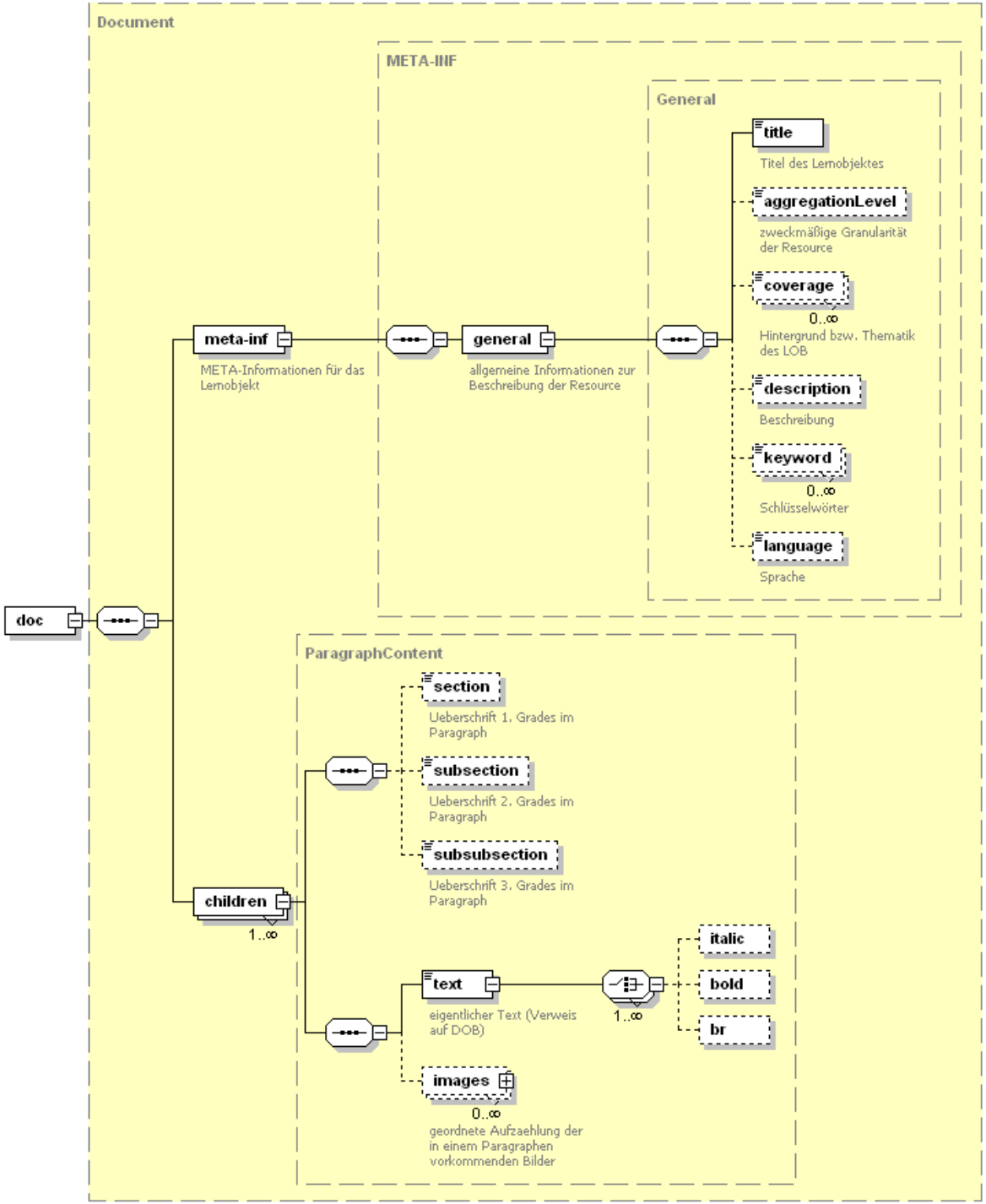
```
<xs:restriction base=„xs:string“>
  <xs:pattern value=„[S][0-9]{6}“/>
</xs:restriction>

<xs:restriction base=„xs:integer“>
  <xs:minInclusive value=„0“/>
  <xs:maxInclusive value=„100“/>
</xs:restriction>

<xs:restriction base=„xs:string“>
  <xs:pattern value=„([a-z])*“/>
</xs:restriction>

<xs:restriction base=„xs:string“>
  <xs:pattern
   value=„[male|female]“/>
</xs:restriction>
```

```
<xs:restriction base=„xs:string“>
  <xs:ennumeration value=„German“/>
  <xs:ennumeration value=„English“/>
  <xs:ennumeration value=„French“/>
</xs:restriction>

<xs:restriction base=„xs:string“>
  <xs:length value=„6“/>
  <xs:whiteSpace value=„collapse“/>
</xs:restriction>

<xs:restriction base=„xs:integer“>
  <xs:totalDigits value=„4“/>
</xs:restriction>
```

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

Reference Schema:

Annotated Document



Download XSD

Schema References

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- *XML in a Nutshell*, 2nd Ed., E.R. Harold, W.S. Means, O'Reilly 2002.
- *Essential XML Quick Reference*, Aaron Skonnard and Martin Gudgin, Addison-Wesley 2001
- XML Schema, a W3C Recommendation from 2 May 2001
  - Part 0: Primer
    http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/
  - Part 1: Structures
    http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/
  - Part 2: Datatypes
    http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

Namespaces

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

Namespaces define different XML vocabularies.
They are used to

- distinguish between elements and attributes from different vocabularies with identical names
- group related elements and attributes from a single XML application for easy recognition

Declaration: `xmlns:<prefix>="<URI>"`

Example: `xmlns:my="http://my.fine.ns"`

Usage: `<my:title>My Joy</my:title>`

Namespaces & Schemas

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

Schemas are designed to support and describe Namespaces.

To associate a (prefixed) namespace with a schema place

```
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace=„http://my.fine.ns“
  xmlns:my=„http://my.fine.ns“>
```

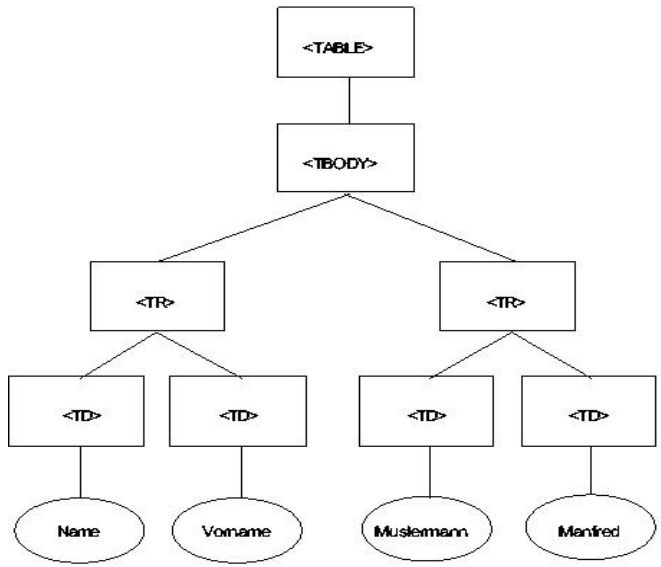
To declare schema described namespace in the instance document:

```
<doc xmlns:xsi="http://www.w3.org/2001/
     XMLSchema-instance"
  xsi:schemaLocation="http://my.fine.ns/
     my.xsd"
  xmlns:my="http://my.fine.ns">
```

Document Object Model

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- General programming interface for HTML and XML-type documents
- Defines the way a document can be accessed and manipulated
- Using a DOM, a programmer can create a document, navigate its structure, and add, modify, or delete its elements
- DOM represents a tree view of the document




```
<TABLE>
 <TBODY>
  <TR>
   <TD>Name</TD><TD>Vorname</TD>
  </TR>
  <TR>
   <TD>Mustermann</TD><TD>Manfred</TD>
  </TR>
 <TBODY>
</TABLE>
```

Java XML Programming (DOM)

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- standardized interfaces for XML processing DOM Level 2, SAX 2.0, XSLT 1.0
- pluggable interfaces for compliant transformers, DOM and SAX parser
- Java Packages:
  - `org.w3c.dom`
  - `javax.xml.parser`
  - `javax.xml.transform`

http://java.sun.com/j2se/1.4.2/docs/guide/xml/jaxp/index.html

Obtaining a DOM Document

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- `org.w3c.dom.Document` is obtained from a `DocumentBuilder`
- A `DocumentBuilder` is created from a `DocumentBuilderFactory` hiding the concrete implementation (► pluggable parsers)
- A `DocumentBuilder` allows either to parse XML content to a DOM document or to create a new one

XML vs. HTML

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- XML was historically designed to exchange data
- XHTML can be defined in XML
- Different designing goals of XML and HTML:
  - XML was designed to describe/structure data and to focus on what data is
  - HTML was designed to display data and to focus on how data looks
  - HTML is about displaying information, while XML is about describing information
  - XML needs a stylesheet definition
  - HTML uses a ‚fixed stylesheet' implemented in the Interpreter (Browser)

XML‘s Impact

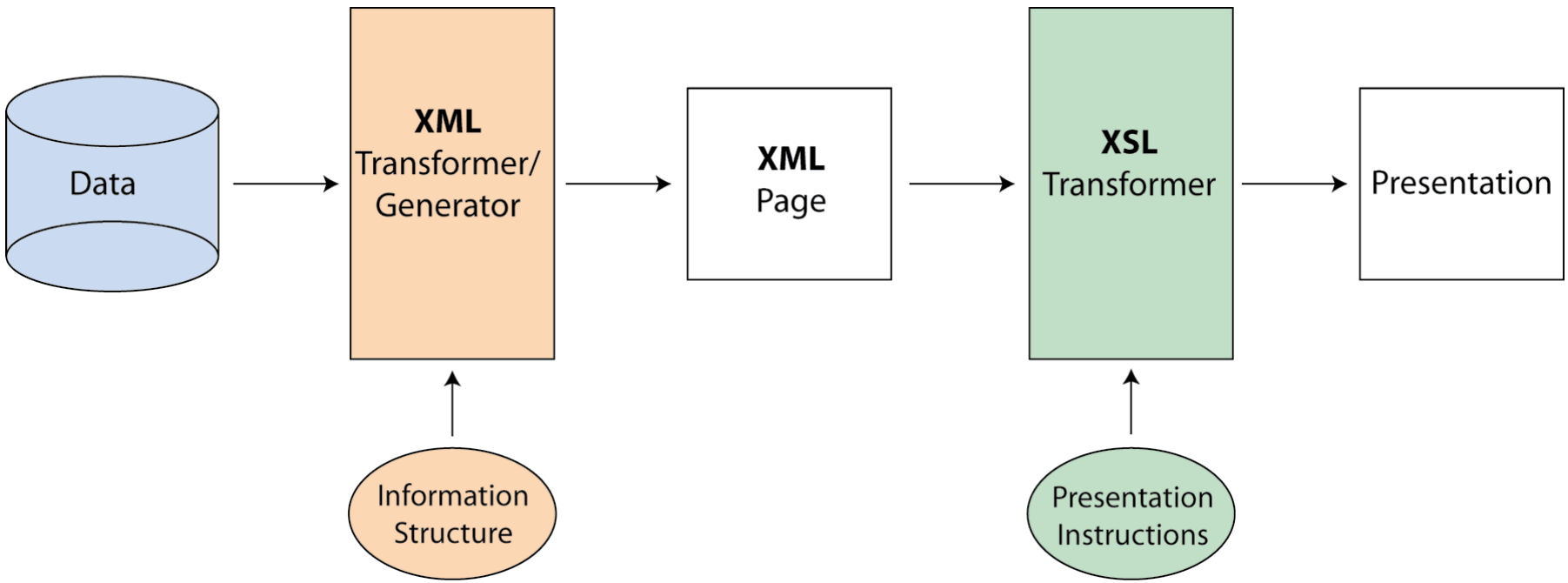
Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- XML opens up the field for defining easy processable markup languages
  - As standards: SVG, SMIL, MathML, RDF, XLINK, ...
  - As customized applications: Schema + Stylesheets
- Any open Web application can be produced by reducing HTML to (one possible) display format
- XML reopens the hypermedia research & development on grounds of open standards

XML Architecture (1)

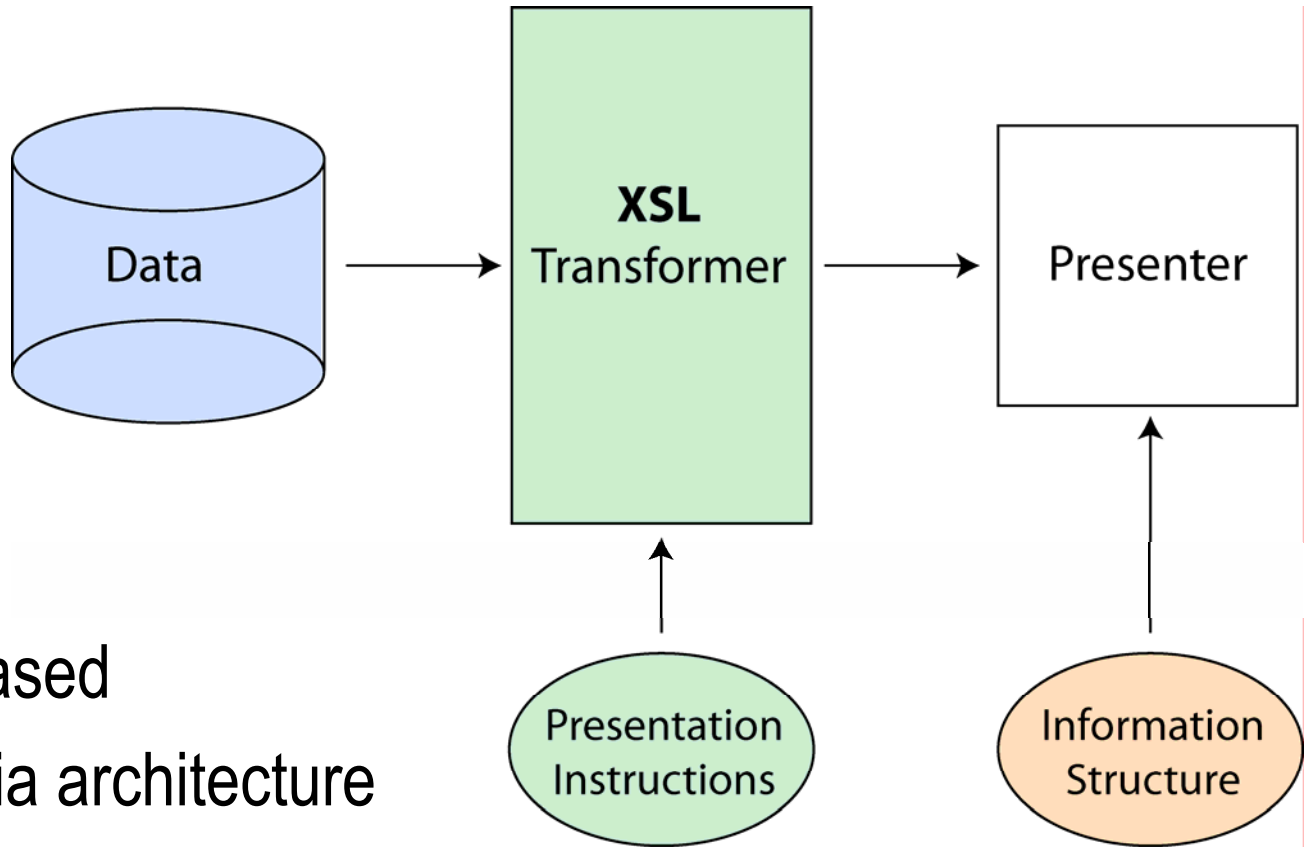
Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

A HAM-type open hypermedia architecture with flexible storage



XML Architecture (2)

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de



A component based
open hypermedia architecture
with XML storage and
linking intelligence at browsers

XPath / XPointer

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- XPath
  - Is a set of syntax rules for graph traversal and selecting elements
  - Was designed to be used by XSLT, XPointer and other XML parsing software
- XPointer
  - Supports addressing into the internal structures of XML documents
  - XPointer defines the meaning of the 'selector' or 'fragment identifier' portion of URIs that locate resources of MIME media types 'text/xml' and 'application/xml'.

Fragment addressing: XPath / XPointer

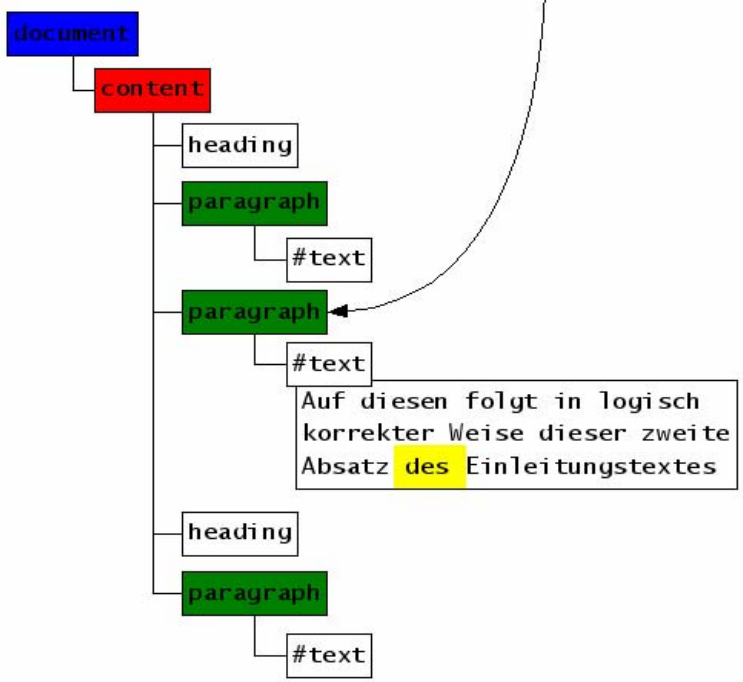
Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

XPointer:

```
xpointer( string-range( /document/content/paragraph[2] ,"des") )
```

XPath:

```
<?xml version="1.0"?>
<document>
    <content>
        <heading>Einleitung</heading>
        <paragraph>
            Das ist der erste Absatz
            des Einleitungstextes.
        </paragraph>
        <paragraph>
            Auf diesen folgt in logisch
            korrekter Weise dieser zweite
            Absatz des Einleitungstextes.
        </paragraph>
        <heading>Kapitel 1</heading>
        <paragraph>
            Der erste Absatz des ersten
            Kapitels setzt das Dokument
            in gewohnter Qualität fort...
        </paragraph>
    </content>
</document>
```



XPath Node Adressing

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

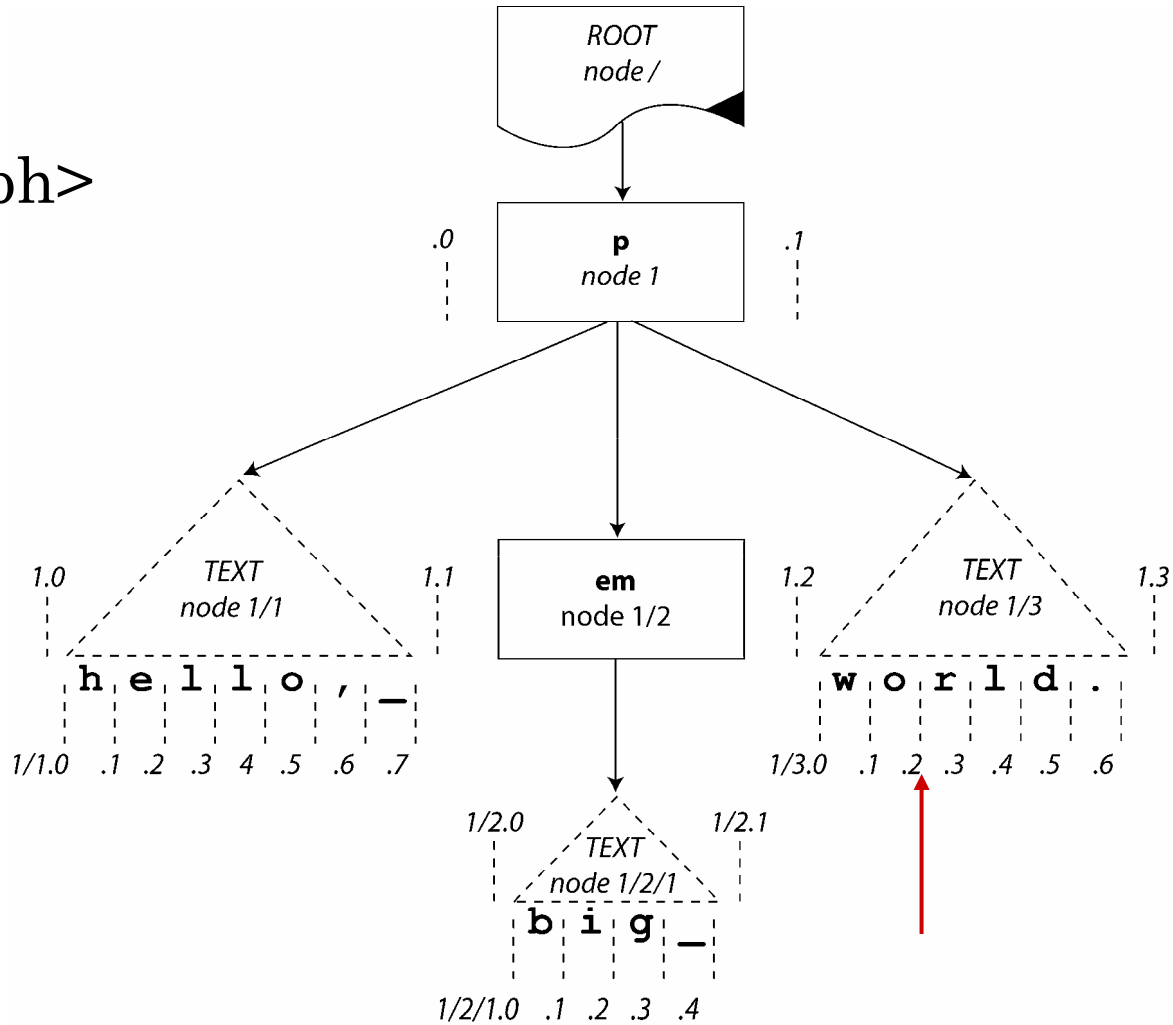
- XPath is used to select element nodes for processing, for conditional processing and for generating text
- Location paths consists of one or more location steps, each separated by a slash
- Location steps, build by an axis, a node test and predicates, selecting a set of nodes
- XPath provides functions (type dependent) for node selection ([starts_with()], [count()], …)
- XML XPath Language (Xpath) 1.0 is a W3C Recommendation on 16 November 1999
  http://www.w3c.org/1999/TR/xpath

XPointer: Linearisation of DOM Tree

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

hello,
<emph>big </emph>
world.

address: 1/3.2



XSL Transformations 1.0

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- XML based programming language
- transform XML documents into other text formats (XML, HTML, XHTML, RDF, SVG, PDF, RTF, CSV, et. al.)
- W3C Recommendation on 16 November 1999
  http://www.w3.org/TR/xslt
- XSLT-Namespace:
  http://www.w3.org/1999/XSL/Transform

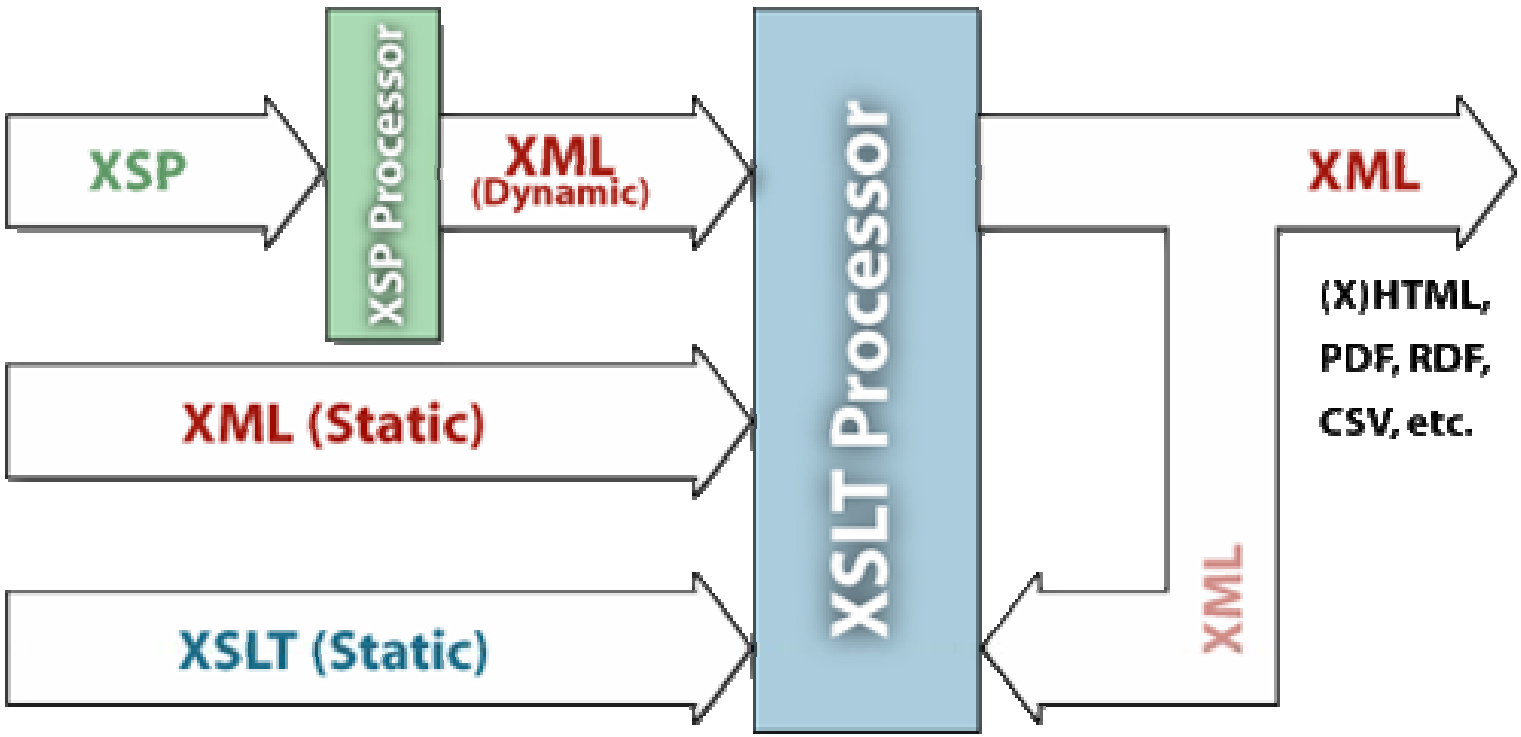
Introduction

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- XSLT describes rules for transforming a source tree into a result tree
- Transformation is expressed as a well-formed XML document
- The result is constructed by finding and processing templates to replace source nodes

XSLT Processing

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de



Programming Models

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- XSLT supports different programming models:
  - Exemplar-based
    - Similar to JSP / ASP approach
  - Procedural
    - Like functional calls in procedural PL
  - Declarative
    - Similar to Prolog, Lisp, Scheme

Exemplar-based transformation

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

– XSLT is used to fill dynamic content in a well-formed XML document template at appropriate locations

XML Document:

```
<doc>
  <meta-inf>
    <title>My Document</title>
  </meta-inf>
  <children>
    <section>Heading 1</section>
    <text>
      <b>Hallo</b><br/><i>Welt!</i>
    </text>
  </children>
</doc>
```

Result:

```
<html>
<head>
  <title>My Document</title>
</head>
<body>
  <h1>Heading 1</h1>
  <b>Hallo</b><br/><i>Welt!</i>
</body>
</html>
```

Stylesheet:

```
<html
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xsl:version="1.0">
<head>
  <title>
   <xsl:value-of
     select="/doc/meta-inf/title"/>
  </title>
</head>
<body>
  <h1>
    <xsl:value-of
      select="/doc/children/section"/>
  </h1>
  <xsl:copy-of
     select="/doc/children/text"/>
</body>
</html>
```

Procedural transformation

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

– Separated and generalized transformation logic in reusable templates can be called like functions

XML Document:

```
<doc>
  <meta-inf>
    <title>My Document</title>
  </meta-inf>
  <children>
    <section>Heading 1</section>
    <text>
      <b>Hallo</b><br/><i>Welt!</i>
    </text>
  </children>
</doc>
```

Result:

```
<html>
<head>
  <title>My Document</title>
</head>
<body>
  <b>Hallo</b><br/><i>Welt!</i>
</body>
</html>
```

Stylesheet:

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
<html>
  <xsl:call-template name="makeHeader"/>
  <xsl:call-template name="makeBody"/>
</html>
</xsl:template>

<xsl:template name="makeHeader">
<head><title>
  <xsl:value-of
     select="/doc/meta-inf/general/title"/>
</title></head>
</xsl:template>

<xsl:template name="makeBody">
<body>
  <xsl:copy-of select="/doc/children/text"/>
</body>
</xsl:template>
</xsl:stylesheet>
```

Declarative transformation

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- Templates are associated to nodes via XPath patterns, starting at the root element
- Each template indicate on which nodes processing is continued

XML Document:

```
<doc>
  <meta-inf>
    <title>My Document</title>
  </meta-inf>
  <children>
    <section>Heading 1</section>
    <text>
      <b>Hallo</b><br/><i>Welt!</i>
    </text>
  </children>
</doc>
```

Result:

```
<result>
  <title>My Document</title>
  <text>
    <b>Hallo</b><br/><i>Welt!</i>
  </text>
</result>
```

Stylesheet:

```
<xsl:transform
 xmlns:xsl=http://www.w3.org/1999/XSL/Transform
 version="1.0">
<xsl:template match="/">
  <xsl:apply-templates select="doc"/>
</xsl:template>
<xsl:template match="doc">
  <result>
    <xsl:apply-templates
      select="/doc/meta-inf/title"/>
    <xsl:apply-templates
      select="/doc/children/text"/>
   </result>
</xsl:template>
<xsl:template match="title">
  <title><xsl:value-of select="."/></title>
</xsl:template>
<xsl:template match="text">
  <xsl:copy-of select="."/>
</xsl:template>
</xsl:transform>
```

Expressions in XSLT

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- XSLT Expressions are build up on patterns which are restricted XPath locations paths
- XPath patterns are used for identifying nodes (select), specifying conditions (if/when) and generating text (value-of)
- Patterns identify “is-a” rather than “have-a” relationships
- XPath patterns are matched against elements in the source followed by a template creating parts of the result

Templates (1)

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- Templates are the basic element defined by XSLT in order to express transformation parts
- The content of a template defines a portion of the result tree
- Templates are invoked for nodes matching approximate patterns, triggered by `apply-templates`
- For modularization aspects, templates can be imported and overwritten

Templates (2)

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- The `match` Attribute contains the expression that is used to select applicable nodes
- The `name` Attribute is used to explicitly call a template in a procedural fashion by `call-template`
- Templates can accept parameters that are declared within the `param` child
- Parameters are filled by `with-param` elements from the calling `apply-template` structure

XSLT References

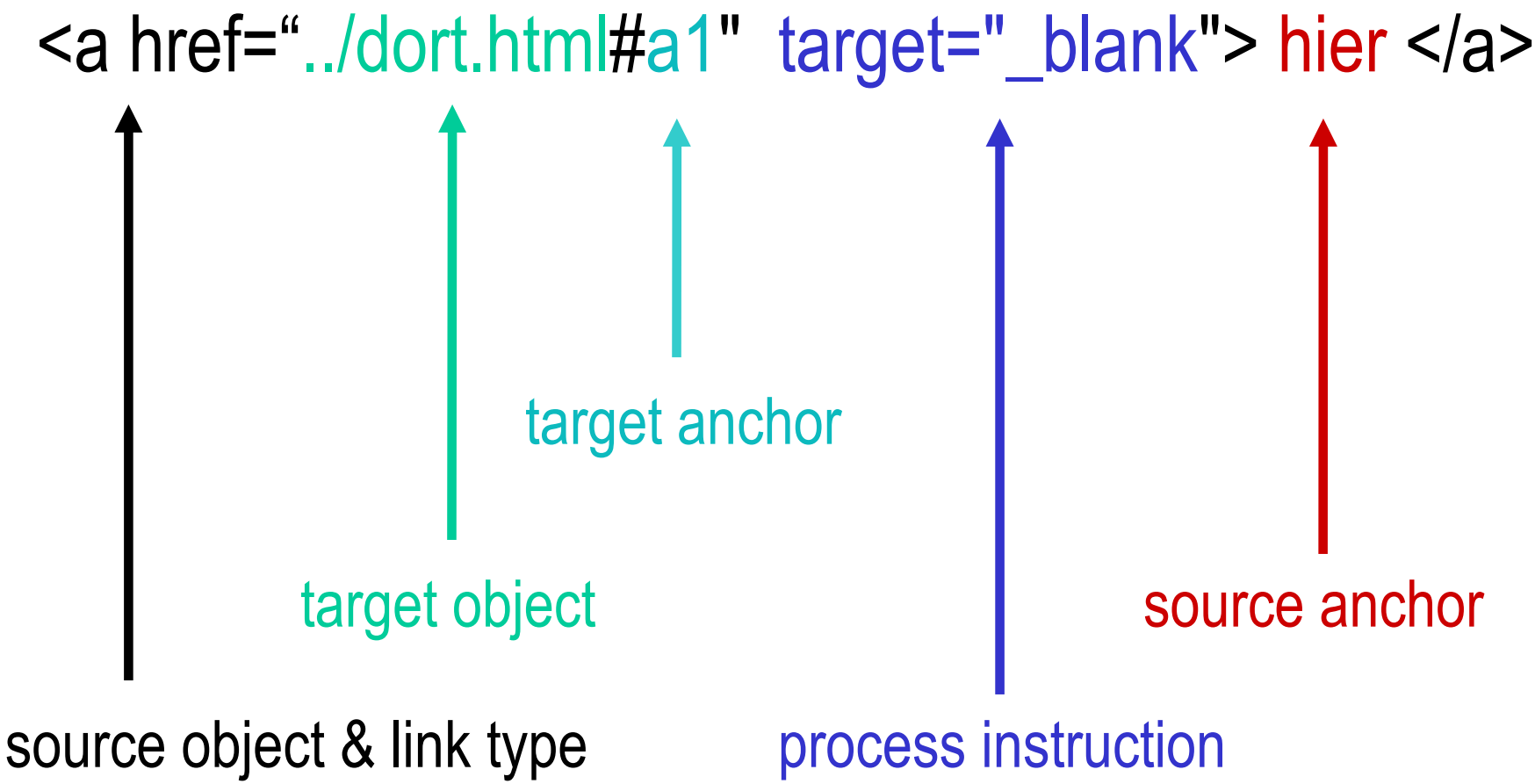
Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- Essential XML Quick Reference , Aaron Skonnard and Martin Gudgi, October 2001, Addison-Wesley
- Xpath Version 1.0 , technical recommendation of the W3C: http://www.w3.org/TR/xpath
- XSL Transformation 1.0 , technical recommendation of the W3C: http://www.w3.org/TR/xslt

Linking & Anchoring

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

Ingredients of a link



Other link types

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- <img src=„mouse.gif" alt=„Mouse">
- <embed src="yippee.wav" width="140" height="60">
- ...

The process instruction is: „follow the link on load“

Hyperreferences construct multimedia applications

Linking Objectives

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- Multivalued / multidirectional links
- Not just ‘Gotos’
- Implicitly defined links
- Temporal links
- Conditional / adaptive links
- Annotation of links
- Dynamic links, computed at runtime
- Collaborative links
- Different ‚linking‘ applied to the same content

⇒ New link models and techniques needed

Embedded vs Non-Embedded Links

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

Embedded

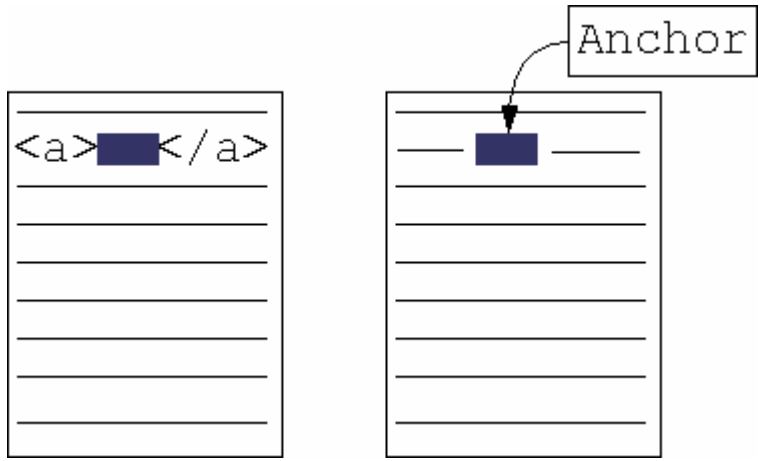
1. Jump address in content
2. Closed [Need special format (HTML)]
3. Mostly text-based media?
4. Often impossible to view all links or automatically process links
5. Simple to distribute
6. Collaboration difficult

External Links

1. Link objects in separate DB
2. Open (Can link to application's own format)
3. Any media type (e.g. video)
4. Easy to view/filter/process all links
5. More complicated to distribute
   - Links are separate from content
   - Separate Link Server needed
6. Collaboration easier
   - No write permission to content needed

Anchoring

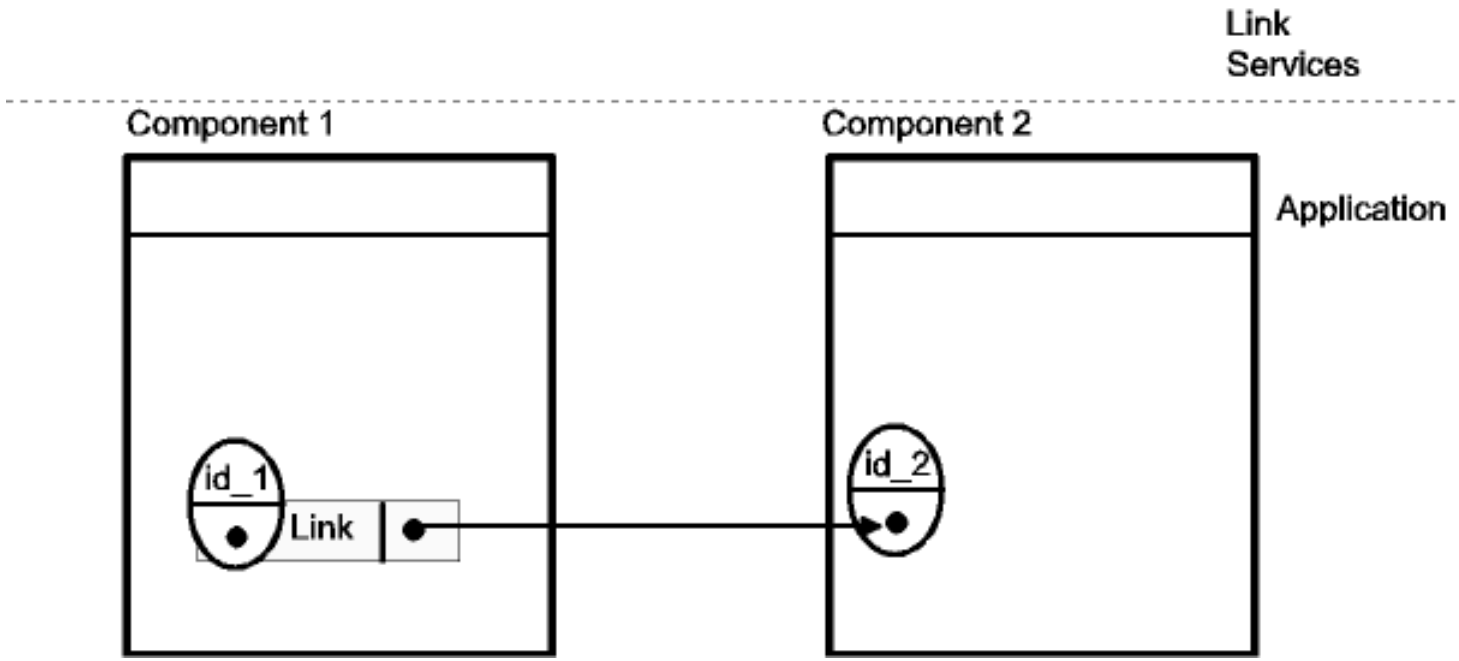
Thomas Schmidt
schmidt@informatik.
haw-hamburg.de



- Anchors are endpoints of links
- Anchors are usually a pair of
  - Address or *local Id*
  - Within-node pointer or *Location Specifier (LocSpec)*
- Anchors are handled differently by different systems
  - Where anchors are stored
  - How the anchor refers to the object within the node data that is to be the physical manifestation of that anchor
- Four types of anchoring mechanisms

1. Embedded Anchors

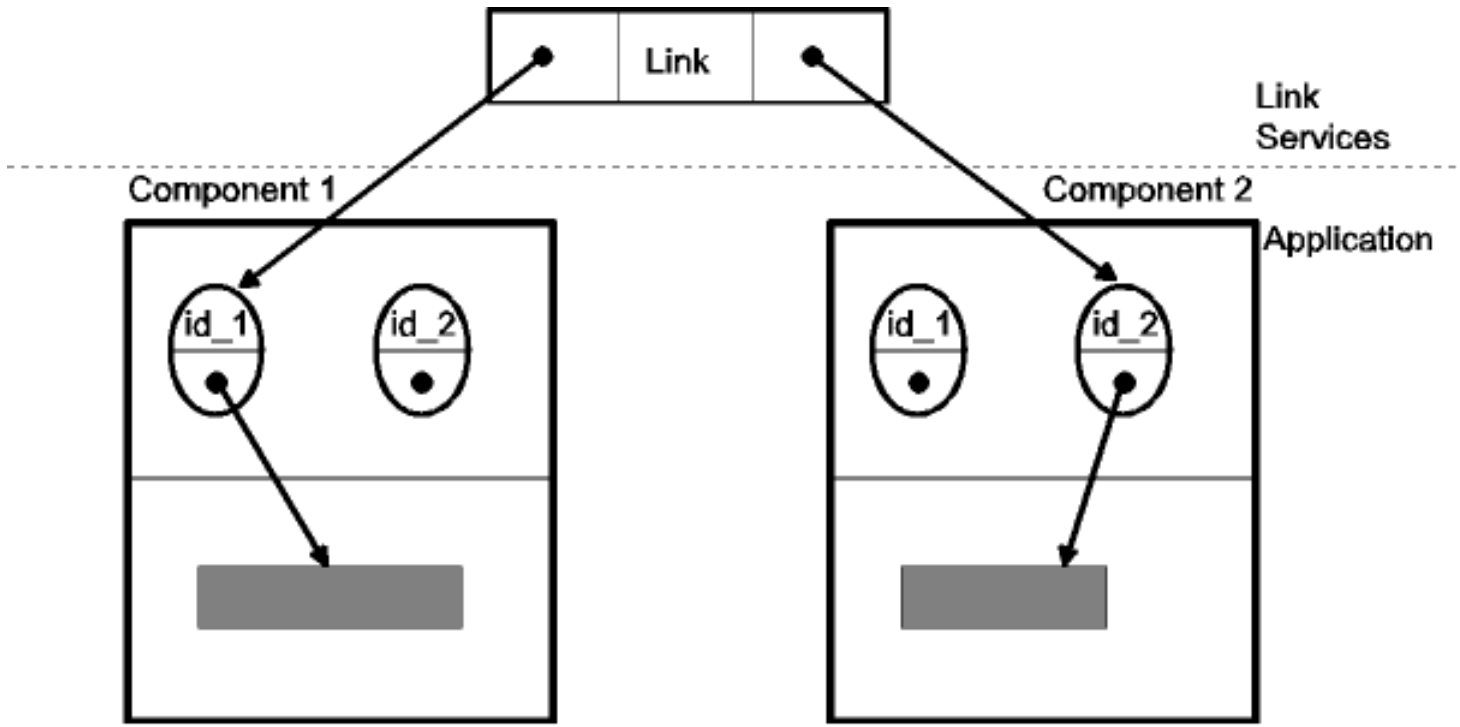
Thomas Schmidt
schmidt@informatik.
haw-hamburg.de



- May also embed links (HTML)
- May carry ID to allocate external link (Hyper-G)

2. Decorating Anchor within Nodes

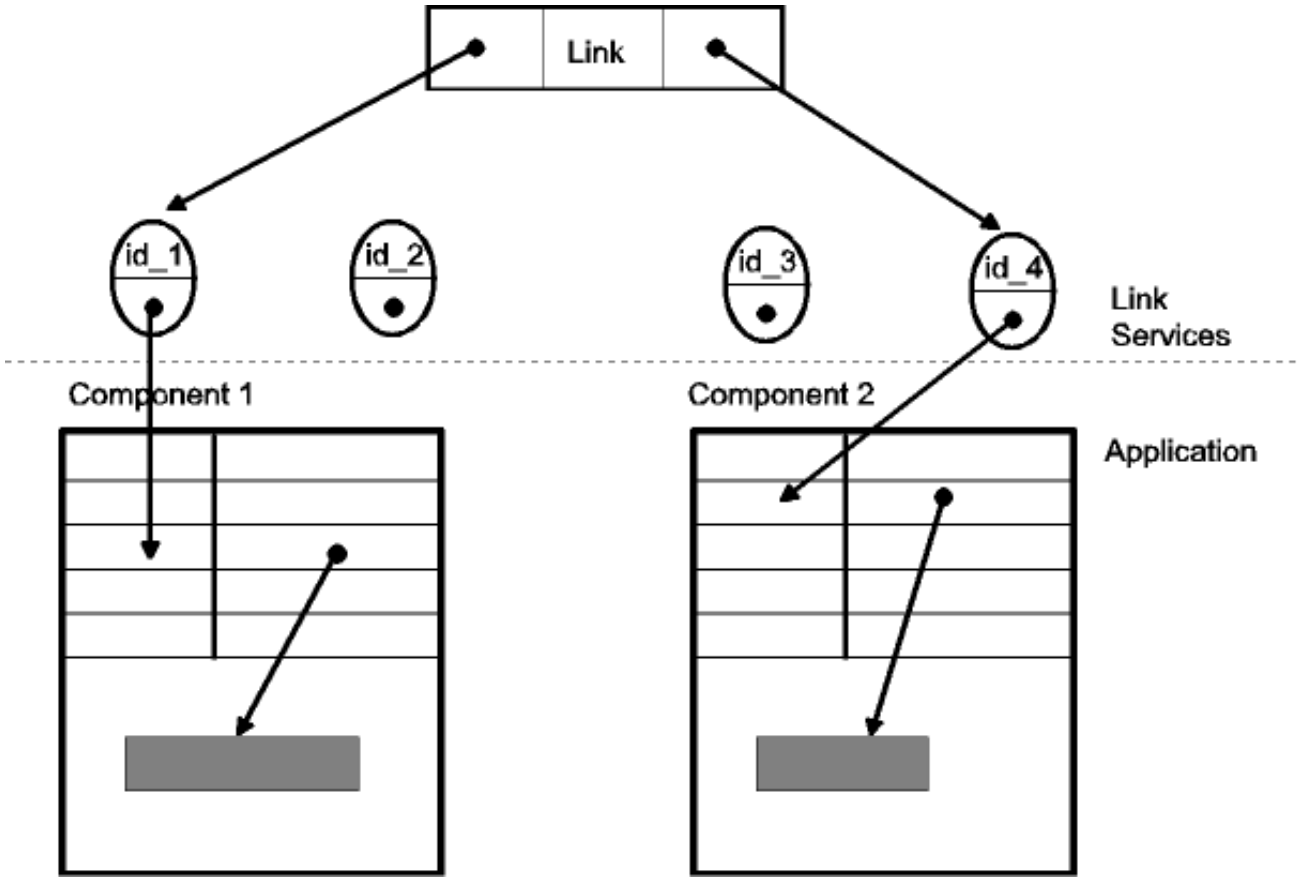
Thomas Schmidt
schmidt@informatik.
haw-hamburg.de



- Dexter type: locally addressable anchors within nodes
- Link Services resolves node + anchor id

3. Global Anchors

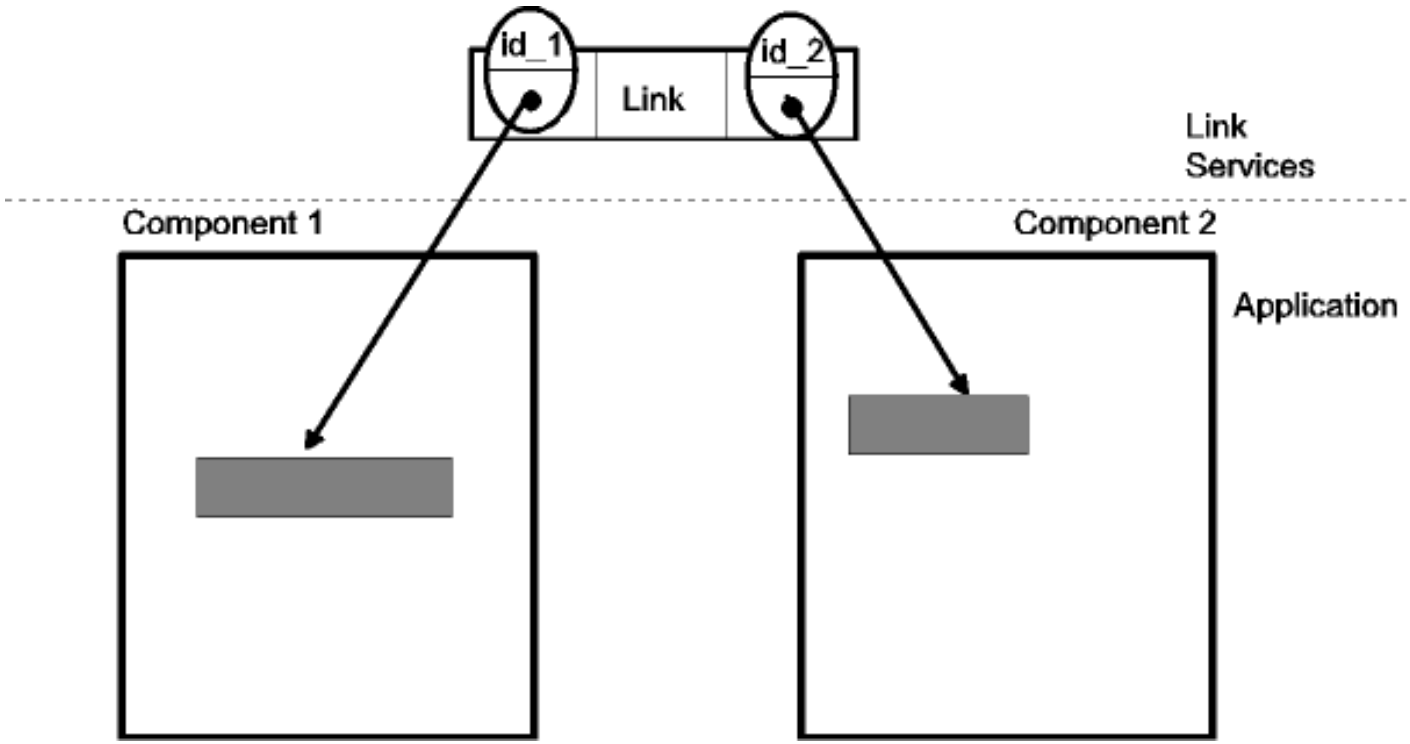
Thomas Schmidt
schmidt@informatik.
haw-hamburg.de



- Hyperbase type: globally addressable decorating anchors
- Link Services handles anchors

4. No Anchors

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de



- Microcosm type: Links address components
- LocSpec transmitted through component interfaces

Fragmenting

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- Addressing „content parts“
- Mime type-specific retrieval of fragments within data units to access data
  - –Text: Xpath/Xpointer (XML)
  - –Vector graphic: Xpath/Xpointer (XML)
  - –Video/audio: SMIL (temporal) / fragments open
  - –Pixel graphic: Imagemap

XLink

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- XML Linking Language 1.0 - W3C Rec. 27 June 2001
- Language standard to express various links between resources
- XLINK Simple Links
  - Structureless outbound links with exactly two participants
- XLINK Extended Links
  - Assert linking relationships among more than two resources
  - Associate metadata with a link
  - Express links that reside in a location separate from the linked resources (linkbases)
- Namespace: http://www.w3.org/1999/xlink

Xlink Attributes

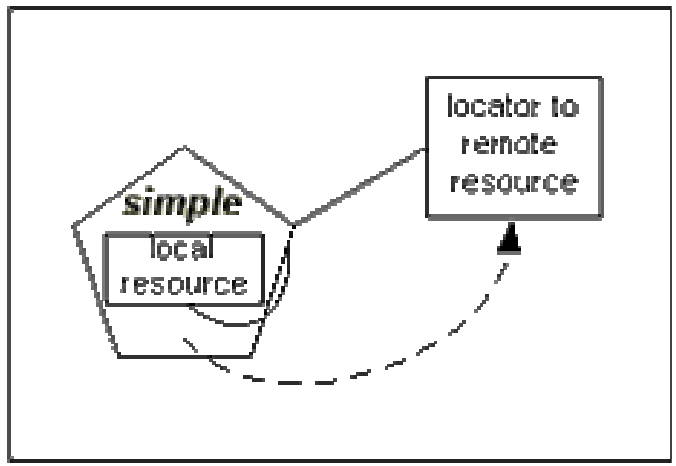
Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- Technical
  - type (of Link)
  - href (locator)
- Traversal
  - label, from, to
- Behavioral
  - show (values: new|replace|embed|other|none)
  - actuate (values: onLoad|onRequest|other|none)
- Semantical
  - role
  - arcrole
  - title
- Extended $\rightarrow$ locator, arc, resource, title

Simple Link

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- HTML-Type Link:



- Bsp:

```
<!ELEMENT studentlink ANY>
 <!ATTLIST studentlink
      xlink:type      (simple)      #FIXED "simple"
      xlink:href      CDATA         #IMPLIED  ...>
<studentlink xlink:href=„www.michael.stud">
                      Michael</studentlink>
```

Simple Link

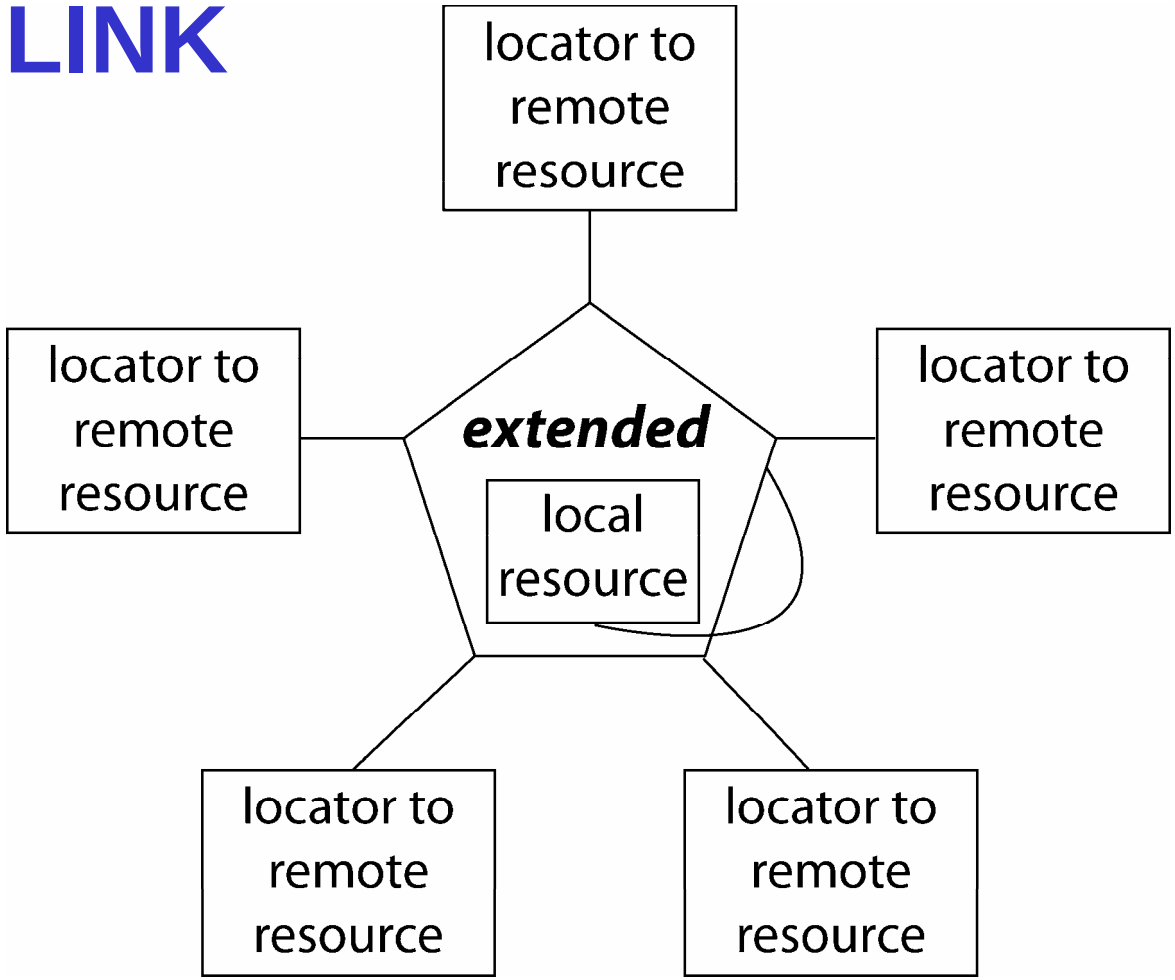
Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- HTML-Type Image Link:

- Bsp:

```
<!ELEMENT studentlink ANY>
  <!ATTLIST studentlink
        xlink:type        (simple)        #FIXED "simple"
        xlink:href        CDATA           #IMPLIED ...>
<studentlink xlink:href=„www.michael.stud/micha.jpg">
                    xlink:show=„embed“
                    xlink:actuate=„onLoad“ </studentlink>
```

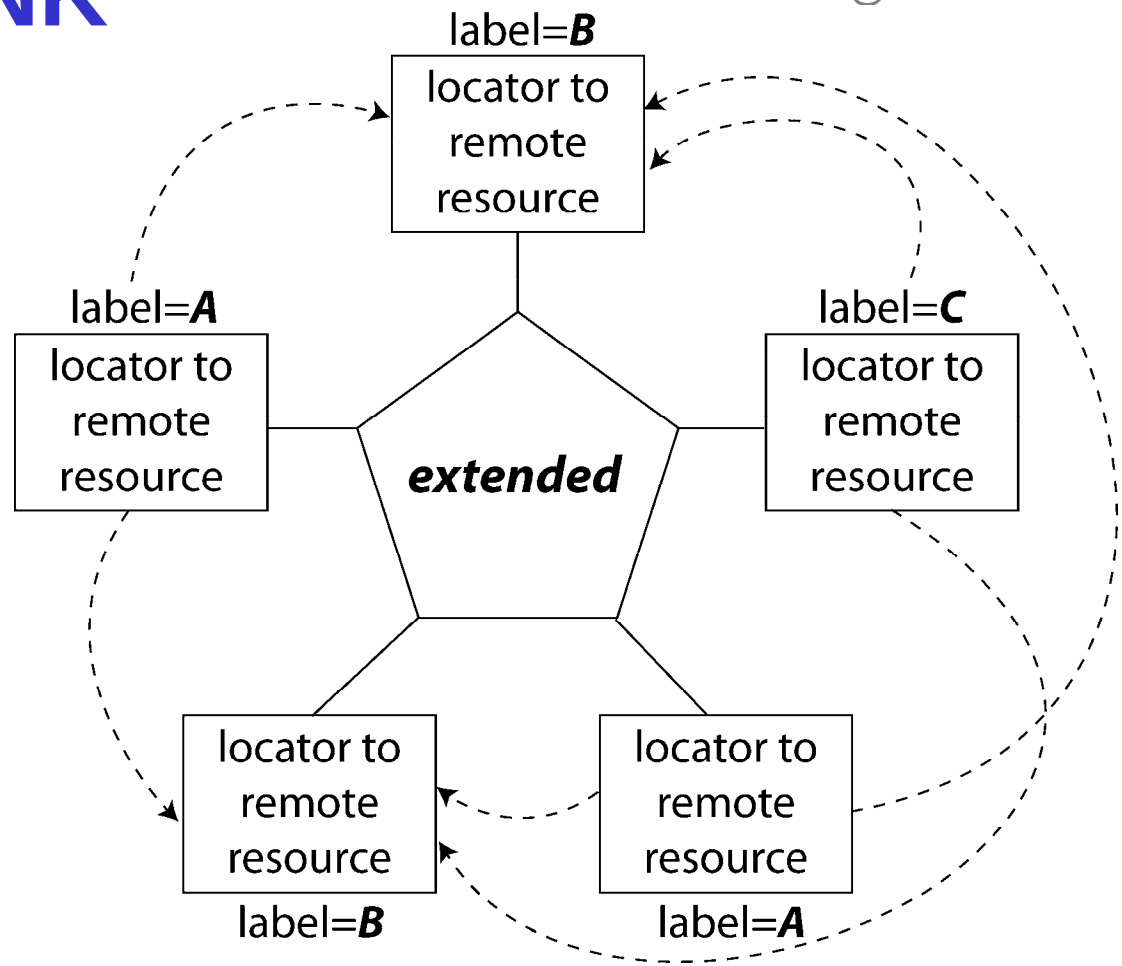
Extended LINK



Multivalued link
between one local resources
and five remote locators

Extended LINK

Thomas Schmidt
schmidt@informatik.



Two arcs defined between labeled Resources:

xlink:type="arc" xlink:from="A" xlink:to="B"
xlink:type="arc" xlink:from="C" xlink:to="B"

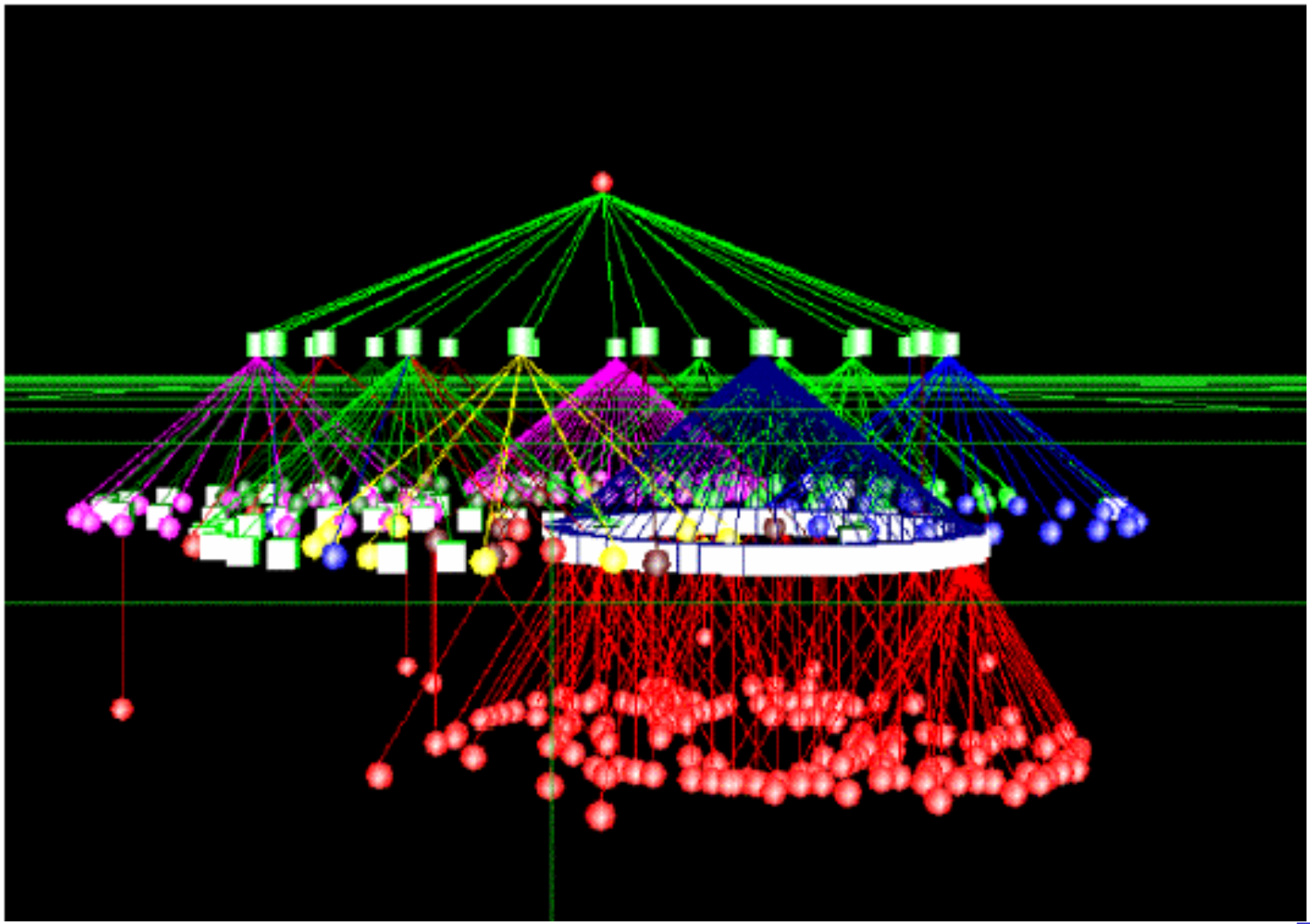
Aspects of Structuring

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- Static and dynamic structures
- Necessary for automated content processing
- Relations between components
  - formal / technical
  - application specific
  - semantical
- Link relation → link consistency

Visualisation: Univit

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de



Hypermedia Authoring

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- Hypermedia authoring is authoring of structures
- Needed: appropriate tools for authoring and maintenance
- Considerations of Hypertext authoring
  - Chunking
  - Interrelationships
  - Contextual access
  - Consistency of documents
  - Simplicity in traversal
  - Screen design
  - Low cognitive load
  - Early reviews
  - Maintain multiple perspectives

Structured Editing

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- Steps of converting existing textual material into hypertext :
  - Splitting the text into portions, which will become nodes
  - Link structures may be derived from existing sources like an index, a table of contents, etc.
  - Objective links can be added, like links from references to theorems or examples to the actual theorems or examples
  - Subjective links are added by the (human) converter who feels they are relevant
  - Links can also be generated between nodes that are similar

Usability

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- Usable systems for writers and readers – both are end-user
- Providing extra tools for information authors
- Authoring tools:
  - Showing links not yet completed
  - Buttom types
  - Graphical overviews
  - Fold-away/verbose display
  - Node lists
  - Structural metrics
- Usability of authoring difficult to measure

Usability Aspects

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- Organization of content
- Organization of navigation paths
- Link effectiveness
- Link differentiation
- Destination prediction

Cooperative Authoring

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- Writing and reading simultaneously by a group of users
- Requires special environments (if the information changes in the background by activities of other user can cause disorientation)
  - Versioning
  - Shared workspaces
  - Additional channels

Concurrency Issues with Collaborative Authoring

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- Several levels of concurrency control
  - Event notification
  - Fine-grained notification
  - User-controlled locking
  - Shared locking
  - Fine-grained locking of persistent collaboration information

Locking

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- User-controlled locking can coexist with locking for short transactions
- No lost of information - While a user is writing a node, the ongoing work will be saved frequently
- User-controlled locking:
  - Shared locking
  - Fine-grained locking
  - Persistent locking

Notification control

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- Allows users to be notified of important actions on the share network of hypertext objects, performed by other users
- Event notification is usually asynchronous
- Synchronous notification requires polling which is very inefficient
- Kinds of notification:
  - Fine-grained Notification
  - Persistent Event Subscription

Transaction control

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- Purpose:
  - Logical units that group operations comprising a complete task
  - Atomicity units whose execution preserves the consistency of the database
  - Recovery units that ensure that either all the steps enclosed within them are executed or none
- Short transaction
- Long transaction

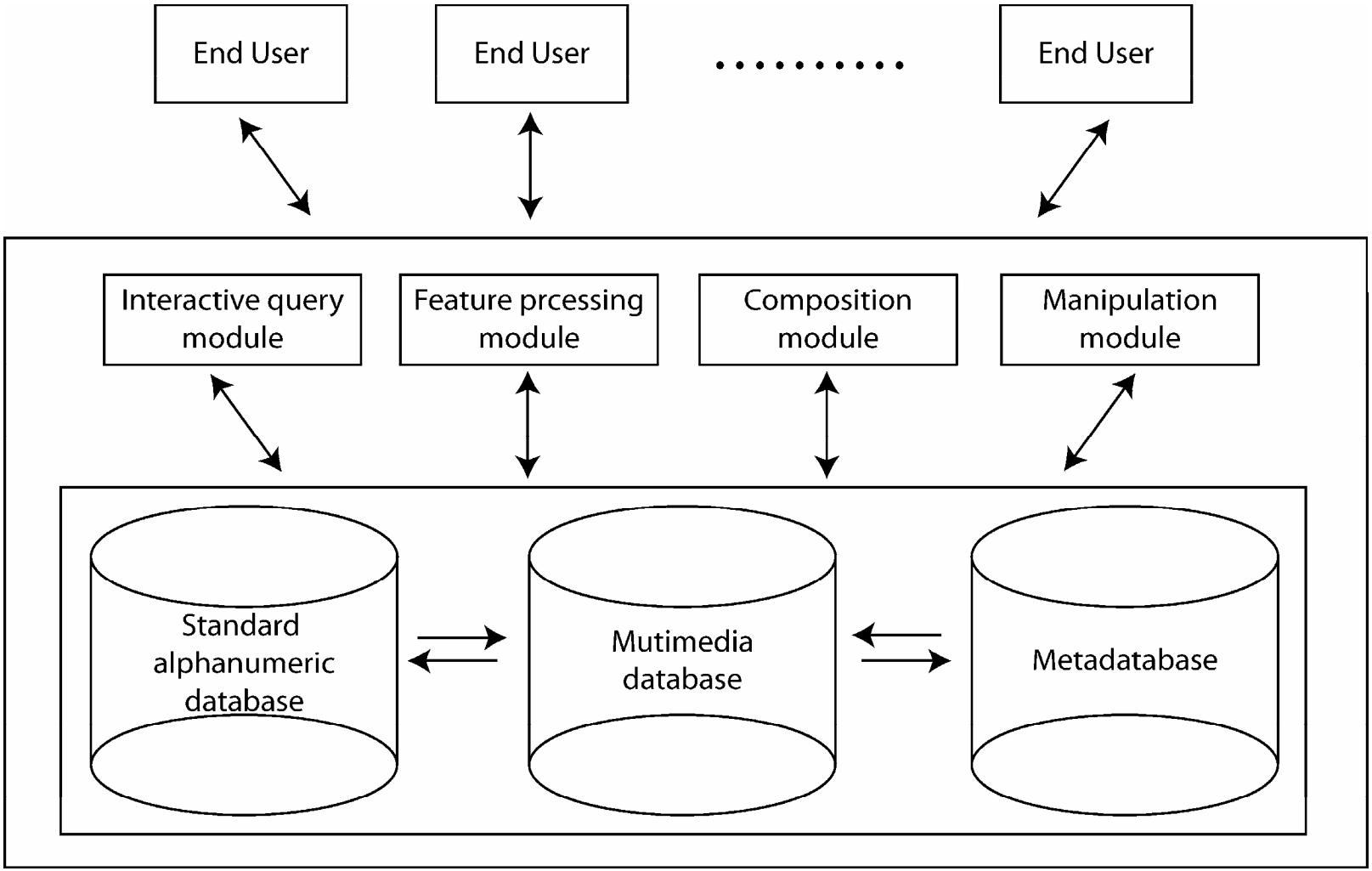
Version control

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- Version control provides the following extensions to concurrency control:
  - Multiple versions of objects
  - Preserves the object identity of different versions of collaborative objects
  - Allows several long updating sessions on the same object
  - Maintaining integrity of individual contributions to an object

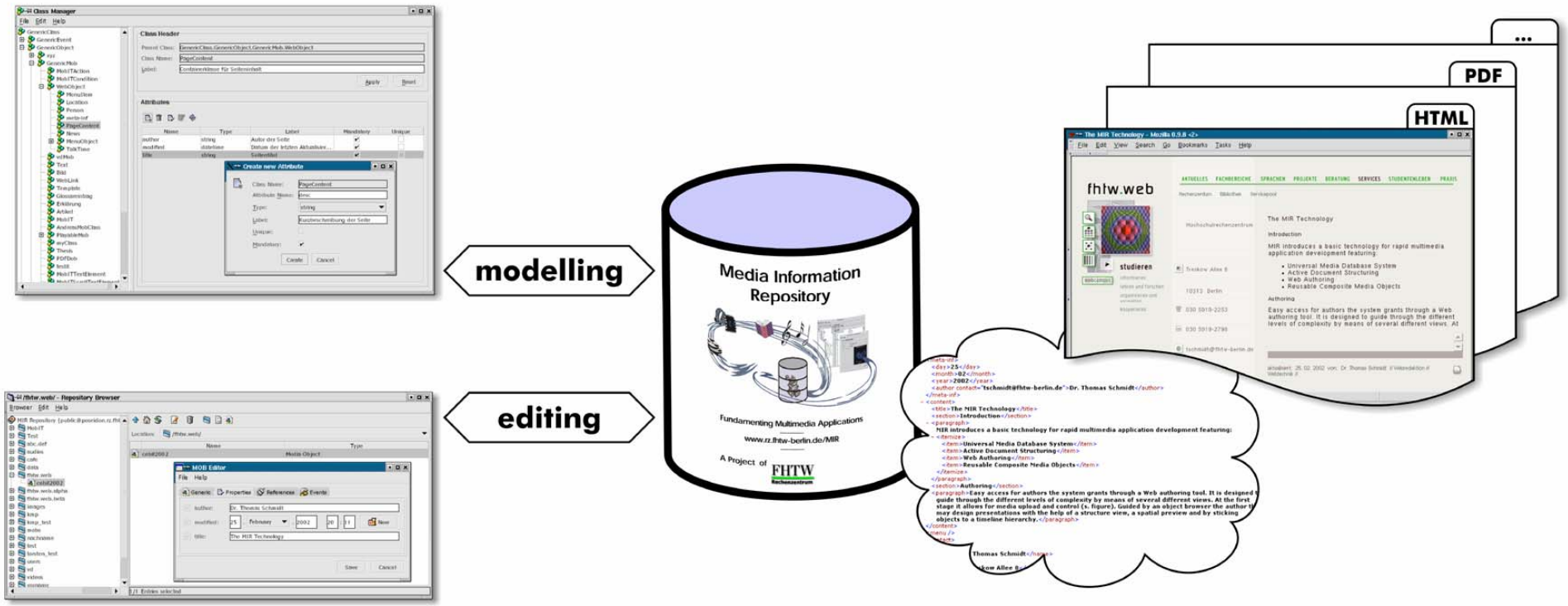
Multimedia Databases

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de



Application Example

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

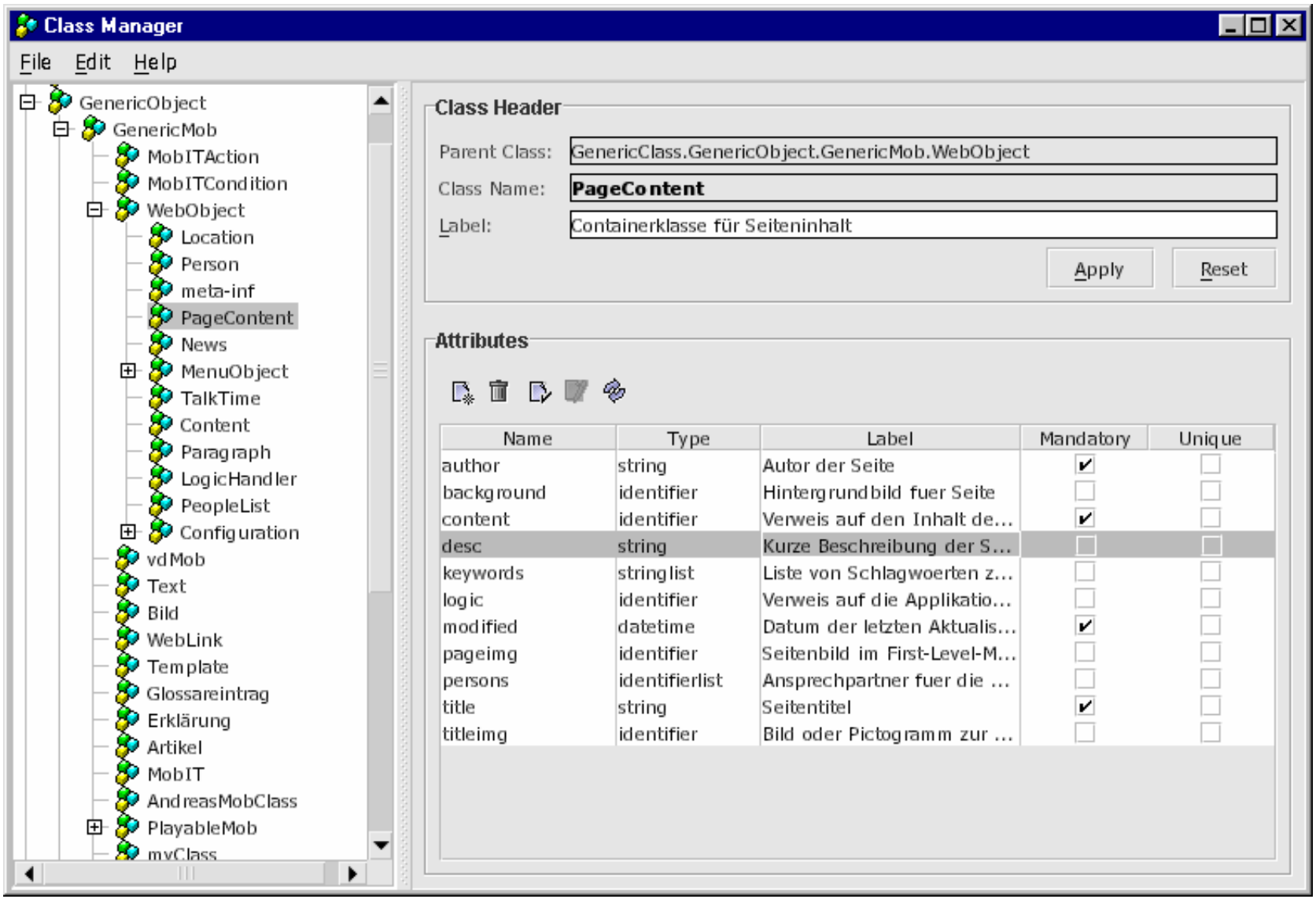


Approach:

Strict Separation of Structure, Content, Logic and Design

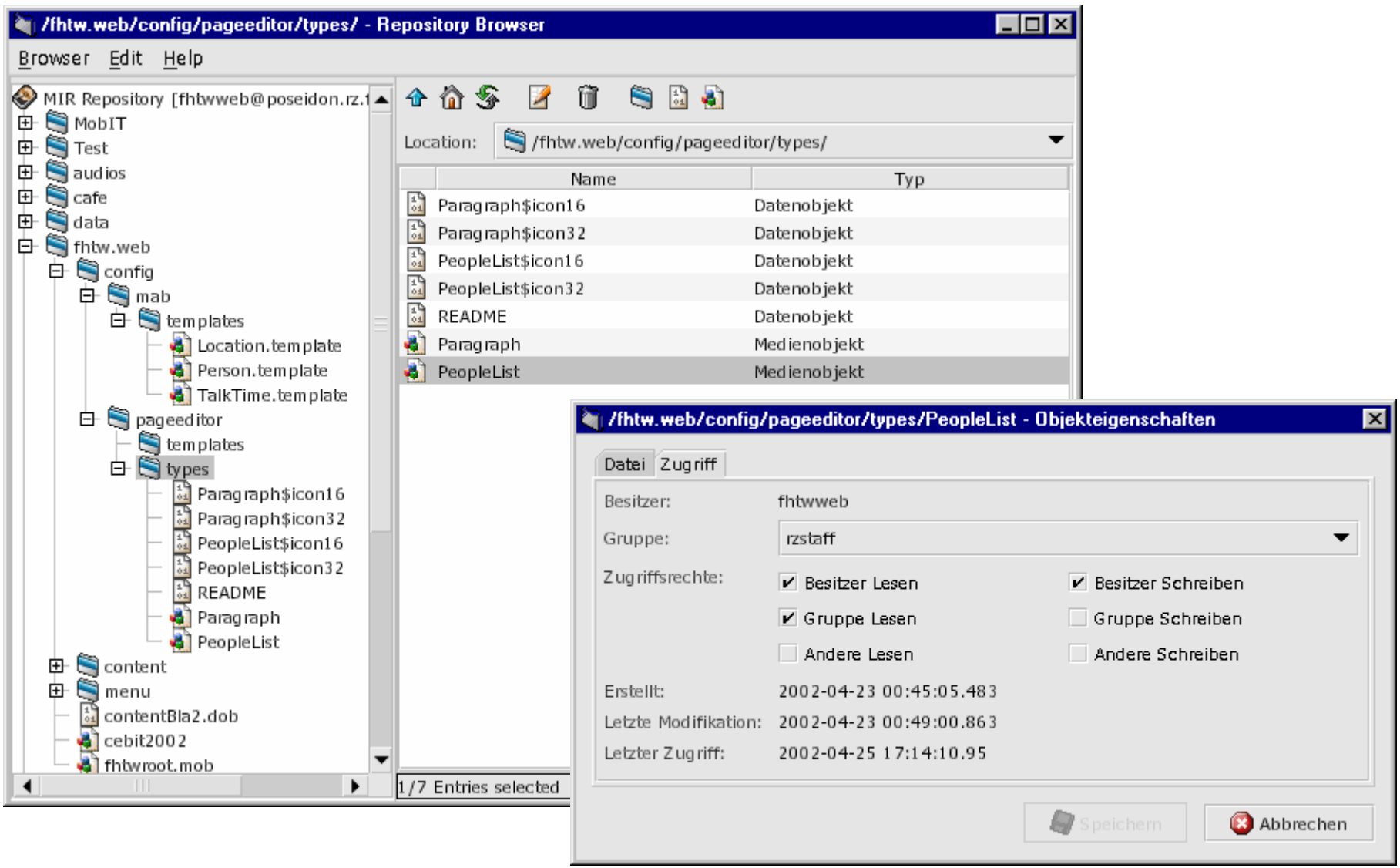
Class Editor

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de



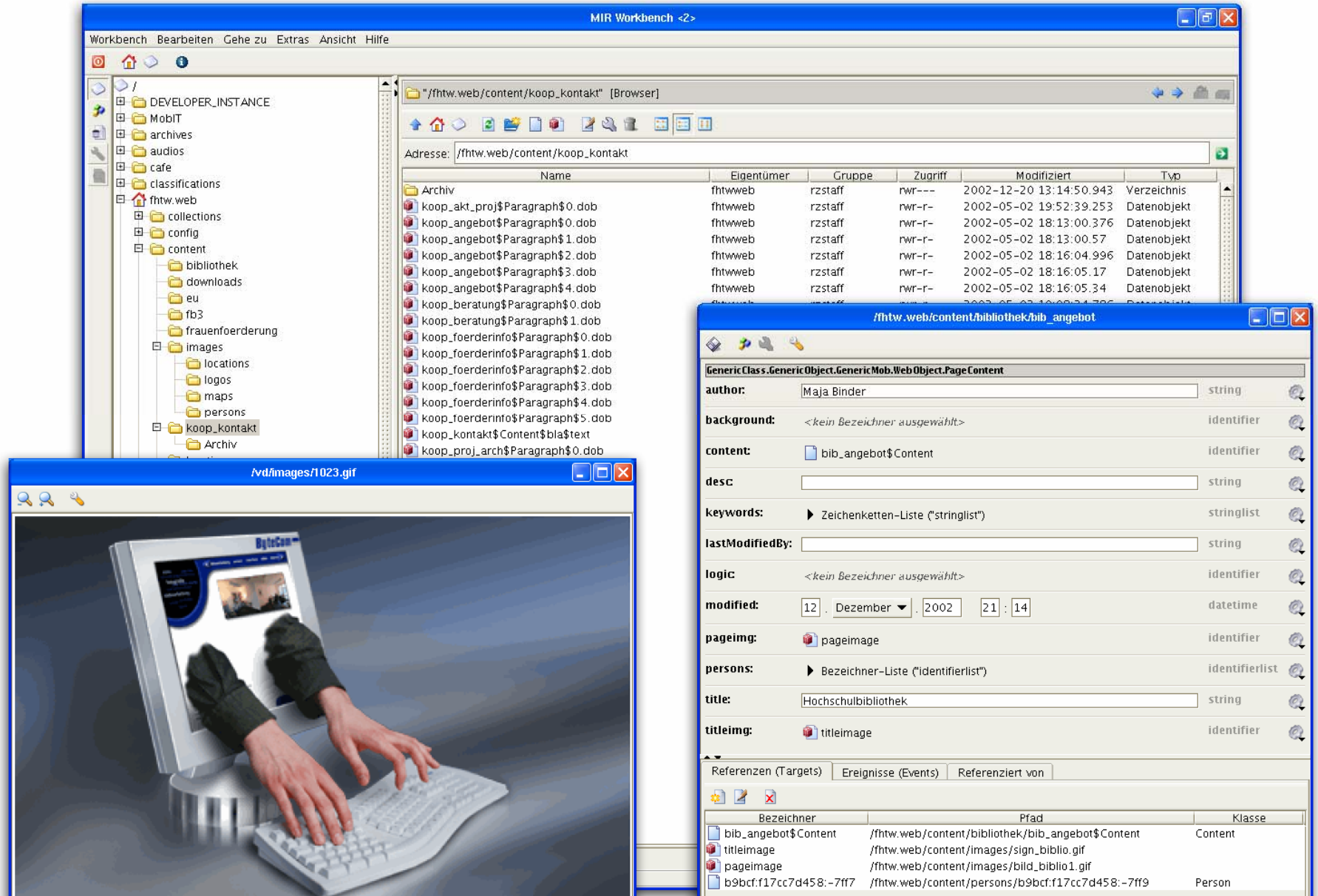
MIR Virtual File System

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de



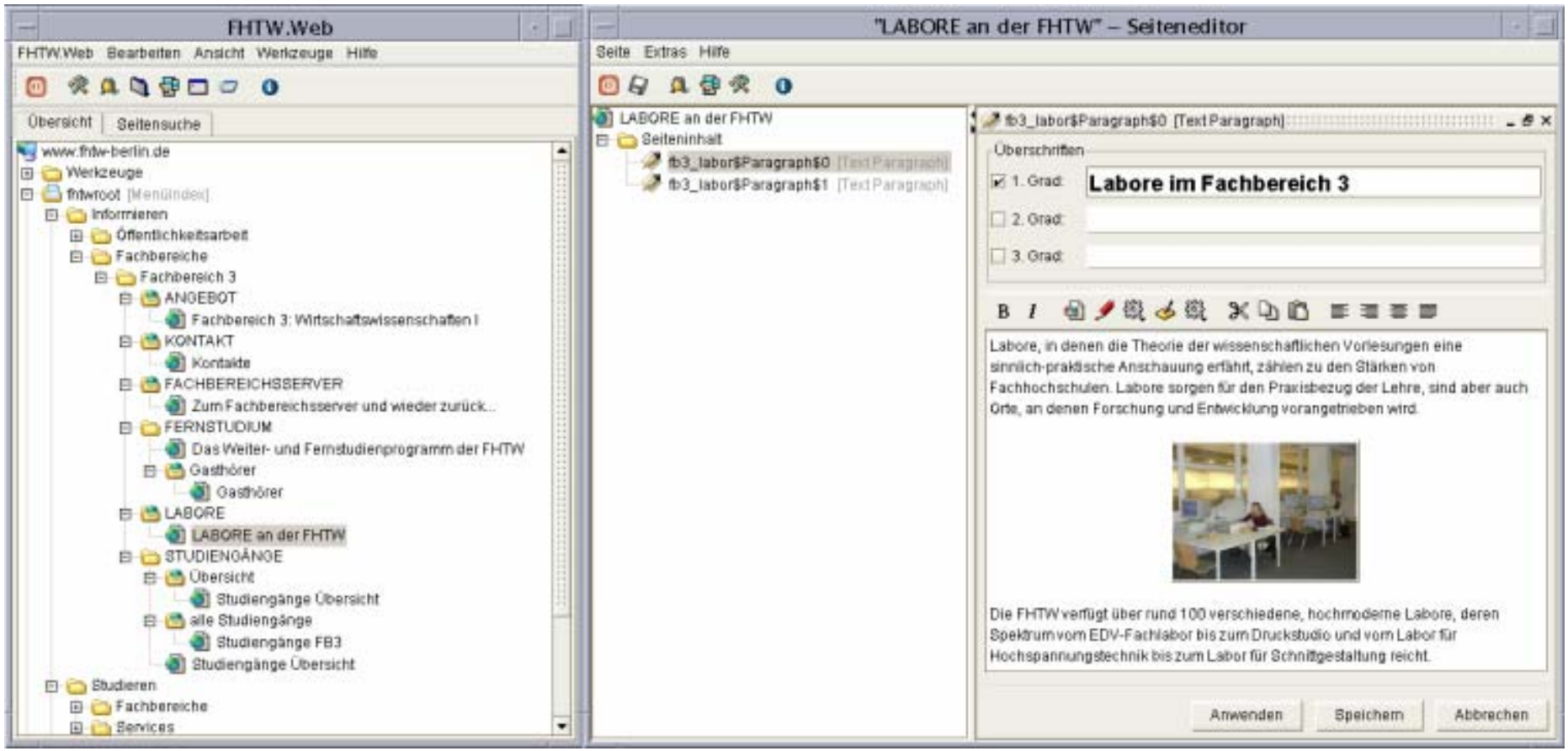
Thomas Schmidt

Authoring - Generic Editors



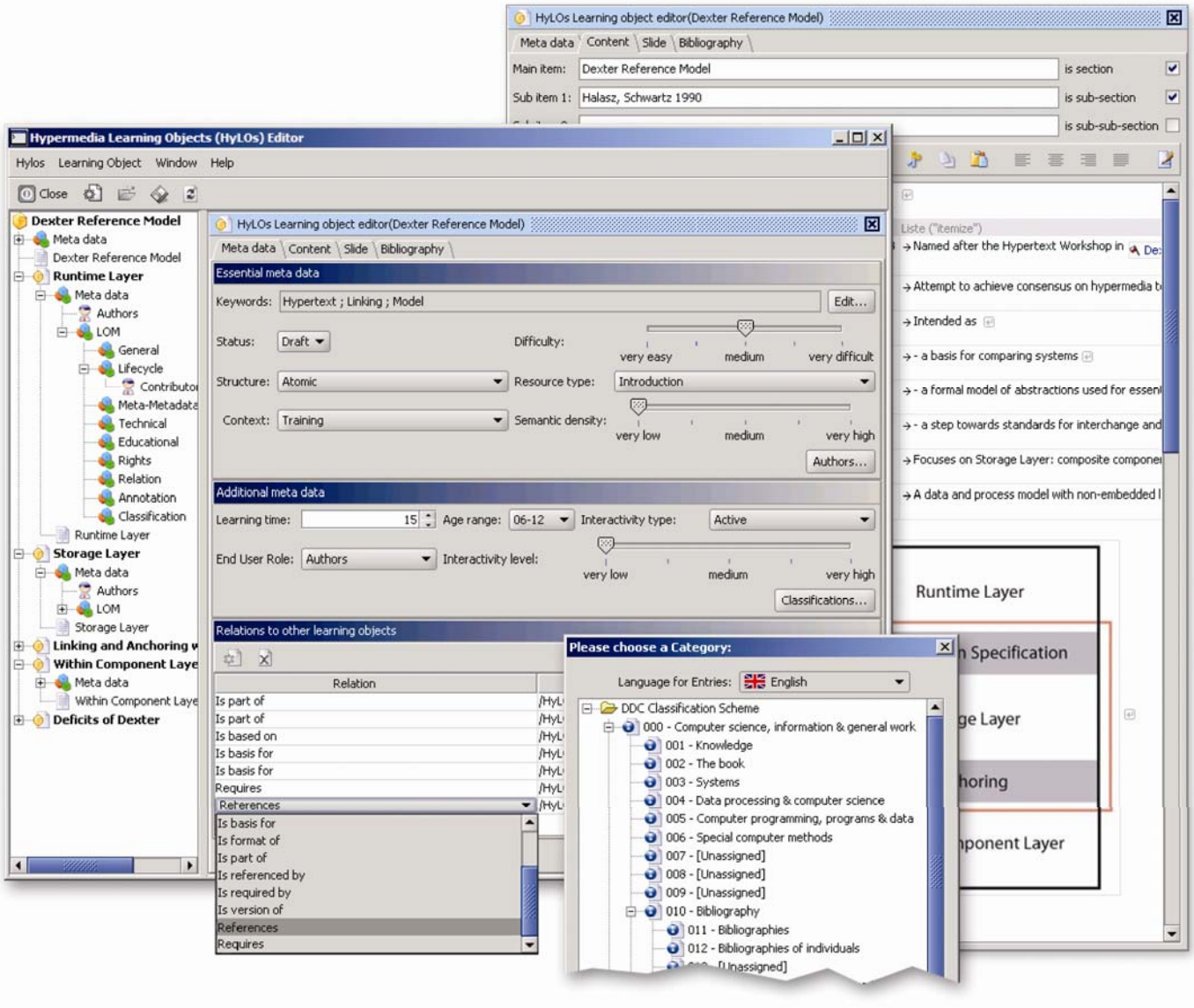
Authoring ‘Pages’ within Navigational Contexts

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de



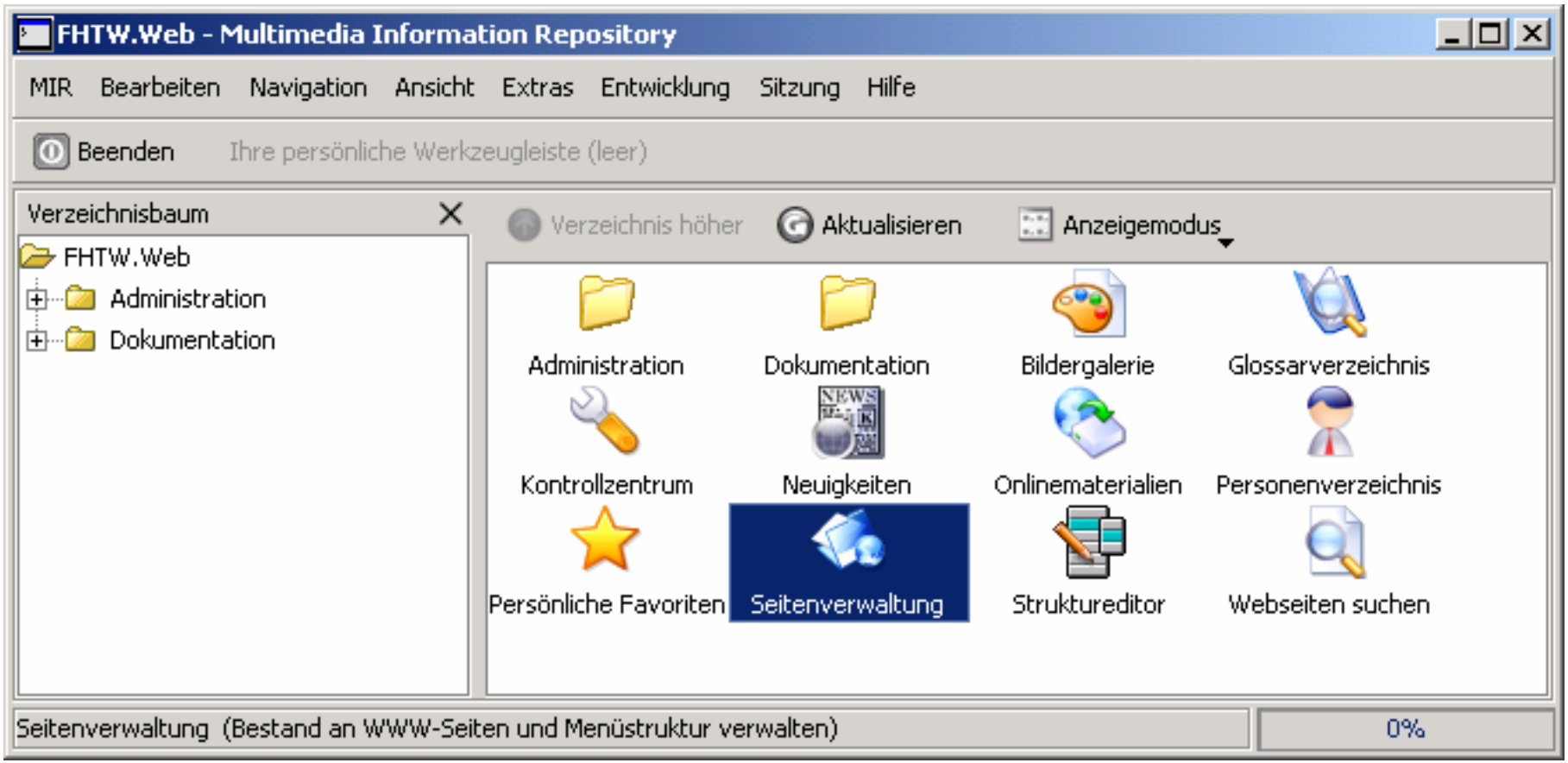
Authoring Complex Information Objects

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de



Authoring Toolbox

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de



Reading

Thomas Schmidt
schmidt@informatik.
haw-hamburg.de

- XML Initiative: http://www.w3.org.
- H. Behne, S. Mintert: XML in der Praxis, Addison-Wesley 2000. Or numerous XML books.
- B. Thuraisingham: XML Databases and the Semantic Web, CRC-Press 2002.