GMD - German National Research Center for Information Technology
Institute for Applied Information Technology (FIT.HEB)
Schloß Birlinghoven, D53757 Sankt Augustin (Germany)

`http://access.gmd.de/`

Accessibility issues of XML and its transformation languages

Dr. Carlos A. Velasco

Carlos.Velasco-Nunez@gmd.de

(2000)

IV Jornadas del SIDAR

1. Introduction

Today's Internet environment:

- **bandwidth demanding applications: streaming media + synchronization**
- **database interaction: e-commerce**
- **different rendering devices: desktop, PDAs, mobile telephony, Bluetooth enabled-devices, etc.**

Solution: Server-side technologies

- **Apache + HTML + PHP + database (MySQL, Oracle)**
- **Apache + XML + XSLT + Java servlets (database)**
- **MS solutions: IIS + ASP + XML**

2. XML - Definitions

Def:

A markup language, based on SGML (ISO-8879)

Why?

- **To extend the capabilities of HTML and simplify SGML**
- **≪To allow SGML to be served, received and processed on the Web≫ (XML rec)**

Where?

- **Extensible Markup Language (XML) 1.0, W3C Recommendation**
  `http://www.w3.org/TR/xml-rec`

3. XML vs. HTML

General problems presented by HTML:

- **HTML is not extensible (by the author, nor the browser manufacturer)**
- **HTML is display-centric (accessibility hurdle)**
- **HTML is not directly reusable**
- **HTML only provides one ≪view≫ of the data**
- **HTML has little or no semantic structure**
- **HTML is not suitable for data exchange**

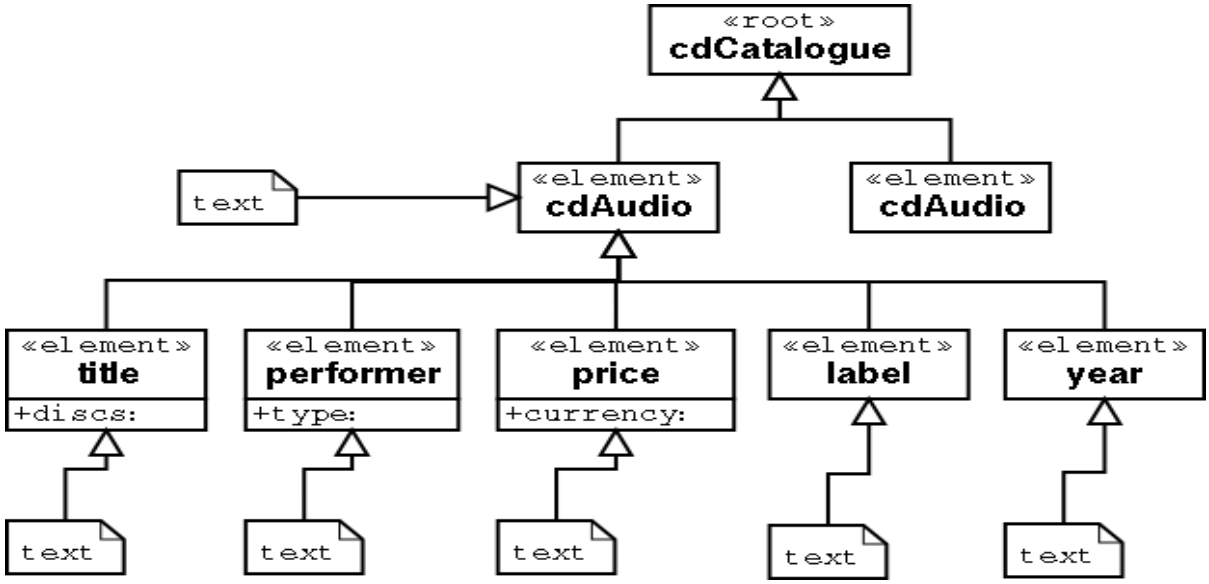
4. XML - Characteristics

- **End of proprietary formats: well suited to interchange information between organizations, individuals and applications (Unicode text)**
- **Information is stored in a hierarchical format (Object Databases vs. Relational Databases)**
- **Strict and consistent syntax ⇒ eases data processing: smart software agents, search engines, etc.**

5. XML - Example

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE cdCatalogue SYSTEM "cdCatalogue.dtd">
<cdCatalogue>
  <cdAudio>
    <title discs="4">Tracks</title>
    <performer type="solo">Bruce Springsteen</performer>
    <year>1998</year>
    <label>Sony</label>
    <price currency="euro">50</price>
  </cdAudio>
  <cdAudio>
    <title discs="1">It&apos;s Only Rock&apos;n&apos;roll</title>
    <performer type="band">The Rolling Stones</performer>
    <year>1995</year>
    <label>Virgin Ben</label>
    <price currency="euro">16</price>
  </cdAudio>
</cdCatalogue>
```

6. XML - Tree model



7. XML - Strength/weakness

Freedom to design new grammars and document types

8. XML - General accessibility recommendations

- **Make the semantics available to the community**
  - **Do not reinvent the wheel. Visit repositories. Approach standardization with the industry**
    - **Schema.net**
      `http://www.schema.net/`
    - **XML.org**
      `http://www.xml.org/xmlorg_registry/`
  - **Document the DTD (accessible format: HTML, text)**

8. XML - General accessibility recommendations (cont'd)

- **Accessible DTD creation:**
  - **Absence of presentation elements: fonts, colors, backgrounds, margins, etc.**
  - **Independence of the rendering technique: enable use of style sheets (CSS2 or XSL + Transformations)**
  - **Provision of alternatives for multimedia content. W3C recommendations (SMIL, SVG, MathML) when feasible**
  - **Define elements to be grouped and indexed (search aid)**
  - **Use of meaningful names for elements and attributes (to be understood without the documentation)**
  - **Define elements with ID attributes to ease navigation**
  - **Inclusion of languages information (attribute `xml:lang` can be defined `IMPLIED`)**

9. XML-based markup languages

W3C extensions to XML:

- **Graphics: SVG**
- **Synchronized Multimedia: SMIL**
- **Mathematics: MathML**
- **Voice rendering: VoiceXML**
- **... more to come**

(Problems with WML/WAP)

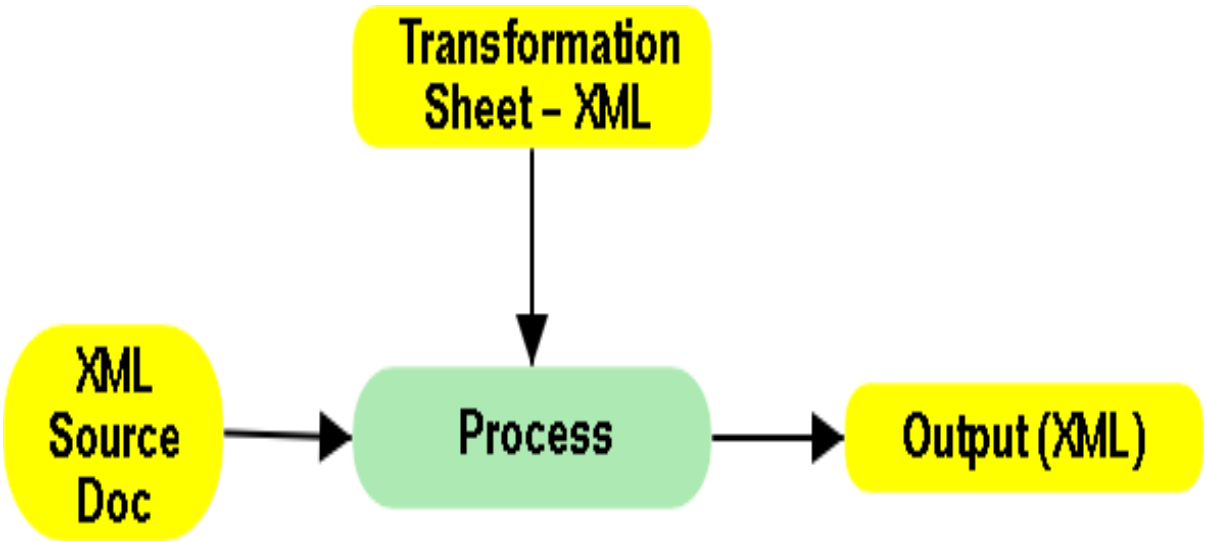
10. XSLT - Introduction

- **XSLT ⇛ eXtensible Stylesheet Language: Transformations**
- **Recommendation ⇛ `http://www.w3.org/TR/xslt`**
- **What is XSLT? “XSLT is a language for transforming the structure of an XML document” (*M. Kay*)**
- **Why transform XML? XML properties**
  - ► **separation of data from presentation**
  - ► **easy transmission of data between applications**
- **Origins in SGML ⇛ DSSSL: Document Style Semantics and Specification Language**

11. XSLT - ≪The family≫

- **XSLT is part of XSL (eXtensible Stylesheet Language): XSL = XSLT + XSL-FO (XSL Formatting Objects)**
- **XPath (sublanguage of XSLT)**
  Rec ⇛ `http://www.w3.org/TR/xpath`
  - **identifies parts of an XML document tree**
  - **allows data manipulation: boolean expressions, string handling, numerical calculations**
- **XML namespaces: coexistence of similar markups (Rec ⇛ `http://www.w3.org/TR/REC-xml-names`)**
- **XML Information Set: essential information of a well-formed XML doc**
  (Rec ⇛ `http://www.w3.org/TR/xml-infoset`)

12. XSLT - How-to



12. XSLT - How-to (cont'd)

Inputs:

- **Multiple XML input docs: `document()` function**
- **Multiple style sheets: `<xsl:include>` or `<xsl:import>` function**

Output document:

- **Another XML document**
- **An HTML document**
- **A text document**

Multiple outputs? Not in the standard; processors

13. XSLT - Properties

A two-steps process:

- **Structural transformation (reminds SQL?)**
  - **Selection**
  - **Aggregation and grouping**
  - **Sorting**
  - **Arithmetical operations**
- **Formatting**

Characteristics:

- **CLOSURE: output and input with "almost" the same data structure**
- **No side effects: progressive rendering**
- **Template (rules) based**

14. XSLT - Example: Hello World!

XML doc:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<greeting>
  Hello, world!
</greeting>
```

Style sheet:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
    <head> <title>Today's greeting</title> </head>
  <body>
    <p> <xsl:value-of select="greeting"/> </p>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```


14. XSLT - Example: Hello World!

Result:

```
<html>
  <head>
    <title>Today's greeting</title>
  </head>
  <body>
    <p>
      Hello, world!
    </p>
  </body>
</html>
```

15. XSLT - Example: CD Catalogue

CD catalogue style sheet:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="cdCatalogue">
<html>
  <head> <title>Our store&apos;s catalogue</title> </head>
  <body>
    <xsl:apply-templates select="cdAudio"/>
  </body>
</html>
</xsl:template>

<xsl:template match="cdAudio">
  <h1>Title description</h1>
  <xsl:apply-templates select="title"/>
  <xsl:apply-templates select="performer"/>
  <xsl:apply-templates select="year"/>
  <xsl:apply-templates select="label"/>
  <xsl:apply-templates select="price"/>
</xsl:template>
```


15. XSLT - Example: CD Catalogue

```
<xsl:template match="title">
  <h2>Title: <xsl:value-of select="."/></h2>
</xsl:template>

<xsl:template match="performer">
  <p> <strong>Performer</strong>: <xsl:value-of select="."/> </p>
</xsl:template>

<xsl:template match="year">
  <p> <strong>Year</strong>: <xsl:value-of select="."/> </p>
</xsl:template>

<xsl:template match="label">
  <p> <strong>Record Label</strong>: <xsl:value-of select="."/> </p>
</xsl:template>

<xsl:template match="price">
  <p> <strong>Price (Euros)</strong>: <xsl:value-of select="."/> </p>
</xsl:template>

</xsl:stylesheet>
```

16. XSLT - Nodes in the Tree Model

- **Root node**
- **Element node: `<tag> ... </tag>`**
- **Text node: PCDATA between tags**
- **Attribute node: within tags `<tag attr="value">`**
- **Comment node: `<!-- ... -->`**
- **Processing instructions: `<? ... ?>`**
  (`<?xml ?>` is not a processing instruction)
- **Namespace node (or declaration)**

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

<prefix:local-name xmlns:prefix="URI-identifier">
```


17. XSLT - Modularity

Style sheet structure is modular:

```
style sheet = style sheet program + style sheet modules
```

- **Instructions:**
  `<xsl:include href="stylesheet2.xsl">` or
  `<xsl:import href="stylesheet2.xsl">`
- **Conflict resolution: parent takes precedence for include; for import, the last definition takes precedence**

18. XSLT - Style sheet in the XML file

The style sheet can be called directly from the XML file: `<?xml-stylesheet ?>` processing instruction.

```
<?xml-stylesheet type="text/xsl" href="../styles/aStyleForMyWeb.xsl" ?>
```


19. XSLT - Accessibility builder?

- **Different processing modes for the document: toc, document, index**
- **Dynamic document rendering, depending on user profiles and preferences**
- **Reuse of code: generators of accessible HTML (WCAG) Example: “a common table processor”**

19. XSLT - Accessibility builder? - Example

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output indent="yes" encoding="iso-8859-1"
  doctype-public="-//W3C//DTD HTML 4.0 Transitional//EN" />

<xsl:template match="/*">
<html>
  <head>
    <title> <xsl:value-of select="@title" /> </title>
  </head>
  <body>
    <table border="1" cellpadding="3">
        <xsl:attribute name="summary">
          <xsl:value-of select="@summary" />
        </xsl:attribute>
      <xsl:apply-templates select="*" />
    </table>
  </body>
</html>
</xsl:template>
<xsl:template match="*">
```


19. XSLT - Accessibility builder? - Example

```
  <xsl:if test="position()=1">
    <tr> <xsl:for-each select="child::*">
      <th>
        <xsl:attribute name="id">
          <xsl:value-of select="name()" />
        </xsl:attribute>
        <xsl:value-of select="name()" />
      </th>
    </xsl:for-each> </tr>
  </xsl:if>
  <tr>
    <xsl:for-each select="child::*">
      <td>
        <xsl:attribute name="headers">
          <xsl:value-of select="name()" />
        </xsl:attribute>
        <xsl:value-of select="." />
      </td>
    </xsl:for-each>
  </tr>
</xsl:template>
</xsl:stylesheet>
```

20. XSLT - Internet Resources

- **World Wide Web Consortium: `http://www.w3.org/`**
- **XMLsoftware: `http://xmlsoftware.com/xslt/`**
- **XSLINFO: `http://www.xslinfo.com/`**
- **The XML Cover Pages: `http://oasis.oasis-open.org/cover/`**
- **XSL-List (Mulberry Technologies): `http://www.mulberrytech.com/xsl/xsl-list/`**

21. Web Publishing Frameworks: Cocoon

- **Selecting a framework: `http://xmlsoftware.com/publishing/`**
  - **Stability - too little beyond 1.x**
  - **Integration with other XML Tools and APIs**
    - **Compatibility**
    - **Follow-up of upcoming standards (W3C)**
  - **Production presence: Real web sites**
- **Cocoon project (funded by Stephano Mazzocchi) `http://xml.apache.org/`**
  - **Xerces: XML parser in Java or C++**
  - **Xalan: XSLT style sheet processor in Java or C++**
  - **Apache FOP (Formatting Objects Processor): in Java**

21. Web Publishing Frameworks: Cocoon (cont'd)

- **How-to? Processing instructions:**

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?cocoon-process type="xslt"?>
<?xml-stylesheet href="URLorFilename" type="text/xsl"?>
...
```

- **XSP (eXtensible Server Pages): Cocoon's technology for building web applications based on dynamic XML content**

22. Conclusions

XML + associated languages will become a ≪de-facto≫ standard. Pros/cons:

- **They represent the end of proprietary formats: data portability**
- **Strong distinction between content and presentation (and even ≪document logic≫)**
- **Freedom for new grammars vs. flexible transformations and reusability**
- **Different scenarios covered by W3C: synchronized multimedia, math, vector graphics, etc.**

23. XSLT - Appendix

XSL-Defined Top-Level Elements

`<xsl:attribute-set>`	`<xsl:key>`	`<xsl:preserve-space>`
`<xsl:decimal-format>`	`<xsl:namespace-alias>`	`<xsl:strip-space>`
`<xsl:import>`	`<xsl:output>`	`<xsl:template>`
`<xsl:include>`	`<xsl:param>`	`<xsl:variable>`

XSL Instructions

`<xsl:apply-imports>`	`<xsl:copy>`	`<xsl:message>`
`<xsl:apply-templates>`	`<xsl:copy-of>`	`<xsl:number>`
`<xsl:attribute>`	`<xsl:element>`	`<xsl:processing-instruction>`
`<xsl:call-template>`	`<xsl:fallback>`	`<xsl:text>`
`<xsl:choose>`	`<xsl:for-each>`	`<xsl:value-of>`
`<xsl:comment>`	`<xsl:if>`	`<xsl:variable>`