Structured Documents on the Web

Jacco van Ossenbruggen

Multimedia and Human-Computer Interaction Group
CWI: Centrum voor Wiskunde en Informatica
Amsterdam, The Netherlands

Presentation Outline:

- Introduction and historical background
- Multiple delivery publishing (MDP)
- MDP & the Web
- Style sheets
- Conclusion

Structured Documents on the Web

Jacco van Ossenbruggen

Multimedia and Human-Computer Interaction Group
CWI: Centrum voor Wiskunde en Informatica
Amsterdam, The Netherlands

Presentation Outline:

- Introduction and historical background
- Multiple delivery publishing (MDP)
- MDP & the Web
- Style sheets
- Conclusion

Documents vs. other data

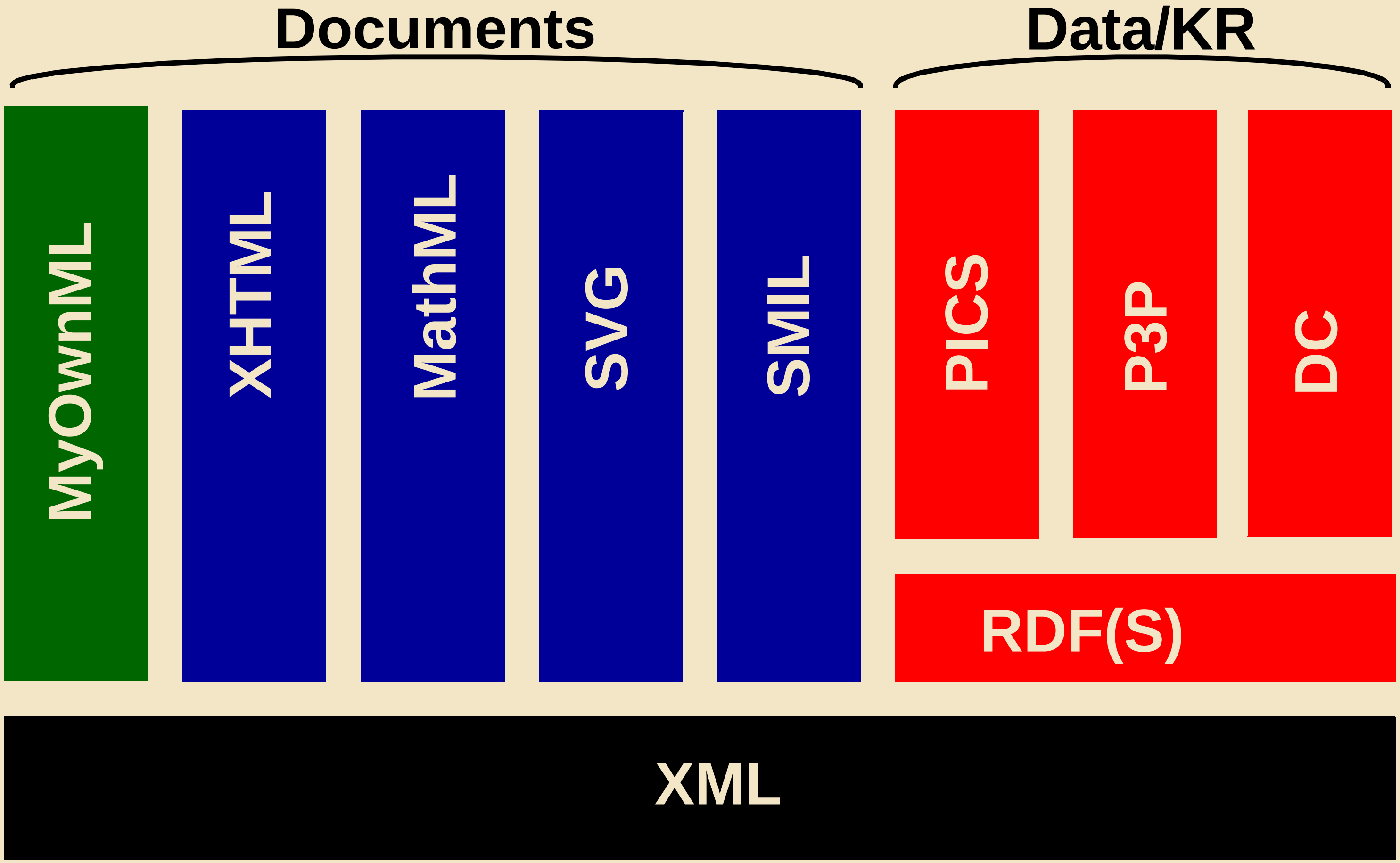
A document is

- a self-contained unit of information,
- intended to be communicated to human interpreter
- examples:
  - book, poem
  - article, paper, report
  - memo, E-mail, letter, etc.

Whereas data can also be

- fragmentary
- intended solely for further machine processing
- examples:
  - database records
  - HTTP requests
  - schemas

Web representation languages



Structured Documents on the Web

Jacco van Ossenbruggen

Multimedia and Human-Computer Interaction Group
CWI: Centrum voor Wiskunde en Informatica
Amsterdam, The Netherlands

Presentation Outline:

- Introduction and historical background
- Multiple delivery publishing (MDP)
- MDP & the Web
- Style sheets
- Conclusion

Electronic Documents

Historically:

- Production electronic
- Dissemination and final-form still on paper
- Goal (final-form): obtain same typographic quality as traditional print
- Goal (authoring): WYSIWYG authoring interfaces (WP,DTP)
  - authoring & storage format mimics final-form presentation format

Currently:

- Both production & dissemination is electronic
- Goal (final-form): exploit presentation possibilities of new media
  - use of audio, video, animation, etc.
  - interactivity (hyperlinks, forms, etc.)
  - dissemination over internet (WWW)
  - use of document technology to access (legacy) information
- Goal (authoring): efficient publication process on industrial scale
  - authoring & storage format differs radically from presentation format

Electronic Documents: Issues

Problem: many document formats cannot cope with changing environment

- hardware dependencies (use of printer/typesetter specific control sequences)
- software dependencies (use of proprietary formats)
- presentation dependencies (layout and style)

Related Issues:

- Longevity (many documents need to last >30 years)
- Maintenance & reuse (c.f. issues in software engineering)
- Flexibility & tailorability (,,   ,,   ,,   )

“Solution”:

- (semi-automatically) convert all documents to new format or new layout
  - expensive & time consuming
  - errorprone (& pretty boring too!)
  - almost always loss of (implicit) information

Real solution:

- multiple delivery publishing model

Multiple delivery publishing (MDP)

MDP distinguishes formats

- one for authoring and long term storage (the “source” format)
- another one final-form presentation (the “target” format)

Needs mapping from source to target format

- mapping can be hard wired into the application
  - e.g. 1st generation HTML browsers
- better: specify the mapping separately
  - such specifications are know as *style sheets*

Source format can now abstract from all details that are likely to change:

- hardware, software, style, ...
- sounds pretty straightforward eh?
- but it actually meant....

Revolution!

Software developers

- no longer control their application’s own file format

Document authors

- no longer control style and layout of their documents

Tools

- no longer used the “sacred” WYSIWYG paradigm

So multiple delivery publishing was/is not obvious at all!

Note: this approach was already advocated by Goldfarb et al. in the 70’s!

- Source documents encoded using IBM’s Generic Markup Language (GML)
- GML was standardized by ISO in 1986 as SGML
- First publicly available parser developed at the VU
  - Amsterdam SGML Parser by Warmer, Van Egmond and Van Vliet (late 80’s)

Multiple delivery publishing & SGML

MDP and SGML remained highly controversial

- People do not like to give up control or change the way they work
- MDP tools could not always match the output quality of more traditional tools
- MDP is no silver bullet!
  - primarily suited for *content-driven* applications
  - not for *layout-driven* applications
- SGML standard is extremely complex
  - still not fully implemented
  - huge and inflexible
  - mainly used in academics and large organizations
  - Netscape’s CEO: “Netscape will never use SGML. Never”

Revival due the World Wide Web

- HTML was an application of SGML (eh... sort of)
- XML is a stream-lined and simplified subset of SGML (really!)
- Published in 1998, XML already has more applications than SGML ever had

Structured Documents on the Web

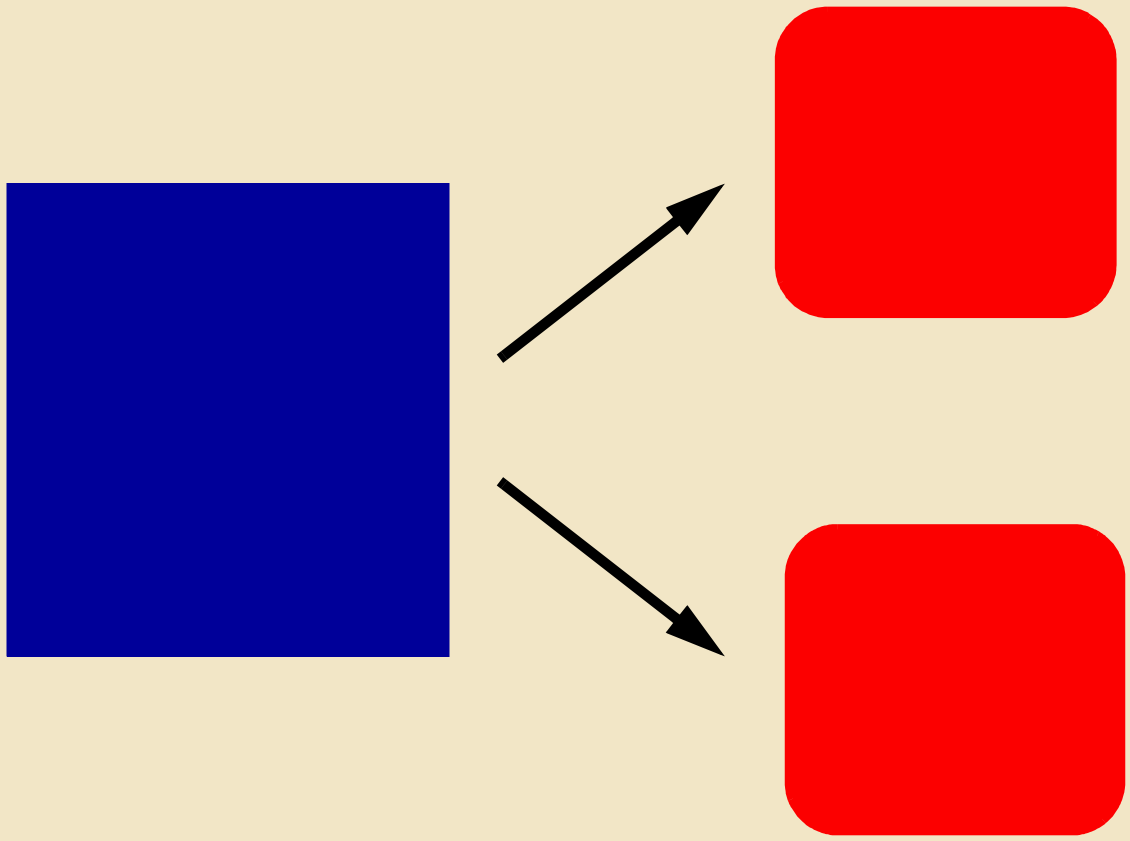
Jacco van Ossenbruggen

Multimedia and Human-Computer Interaction Group
CWI: Centrum voor Wiskunde en Informatica
Amsterdam, The Netherlands

Presentation Outline:

- Introduction and historical background
- Multiple delivery publishing (MDP)
- MDP & the Web
- Style sheets
- Conclusion

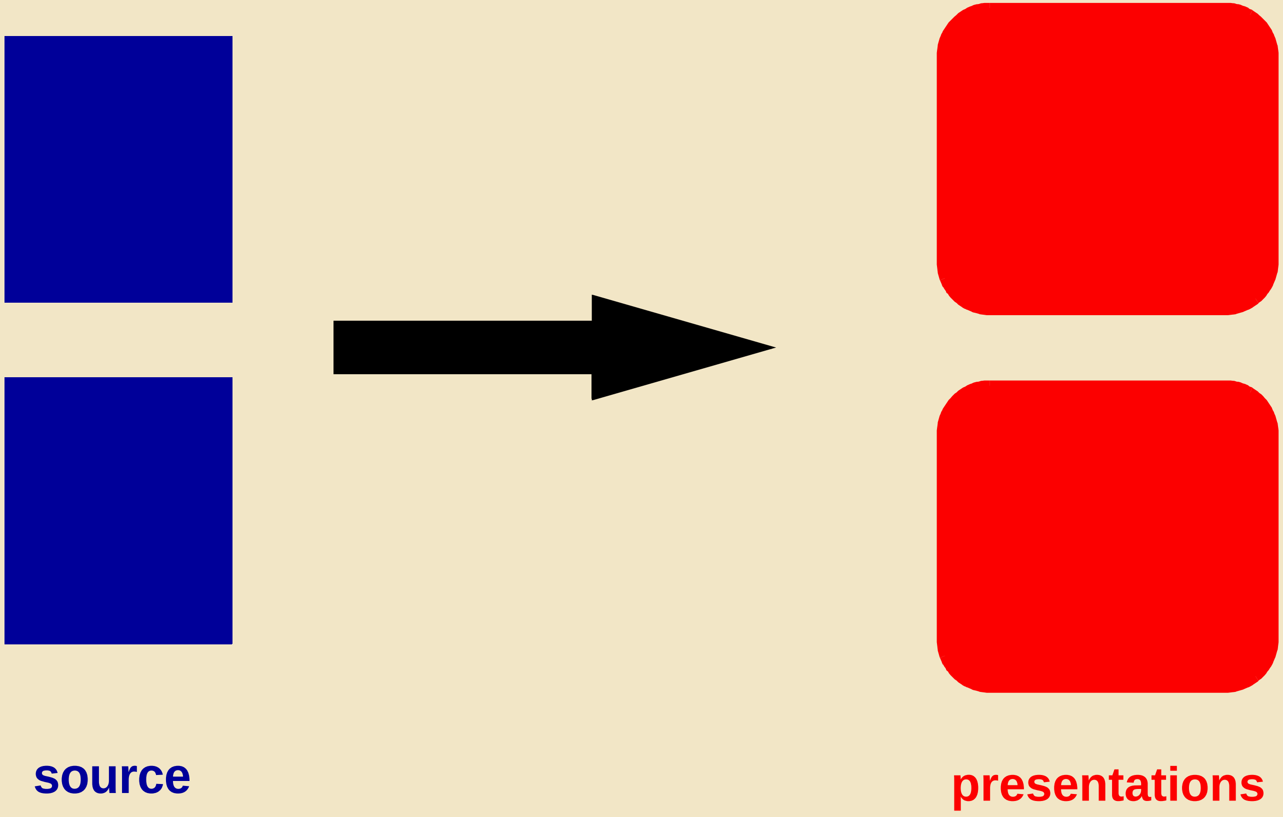
MDP: easy reuse of source document



source document

target presentations

MDP: easy reuse of style specification



Issues for source & presentations formats

Think about design dimensions such as:

- Content versus markup
  - what is in the tags, what is between the tags?
- Embedded versus external markup
  - what is encoded in the same file, what is stored elsewhere?
- Declarative versus procedural
  - specify what or specify how
- Domain independent versus domain specific
  - **`<title>`** or **`<product-shelf-number>?`**
- Layout-driven versus content-driven applications
  - magazine cover or technical manual?
- Visual markup versus structured markup
  - **`<i>`** or **`<emph>?`**

Source vs. presentation format

Source format:

- Structured, declarative markup
- Can be domain independent but...
- ...is usually tailored to a specific domain
- Provide sufficiently rich structure for style sheets and other processing

Presentation format:

- Visual, often procedural markup
- Can be platform/medium independent but...
- ... is usually tailored to a specific output medium/device
- Provide sufficient information to obtain high quality output

How do you classify your favorite document format?

Domain independent vs. domain specific

Domain independent

- Examples: HTML, Docbook, (LaTeX)
- Wide deployment: easy to learn, many (cots) tools available
- Weak semantics for automatic processing other than presentation
- Tools only need to deal with predefined markup semantics

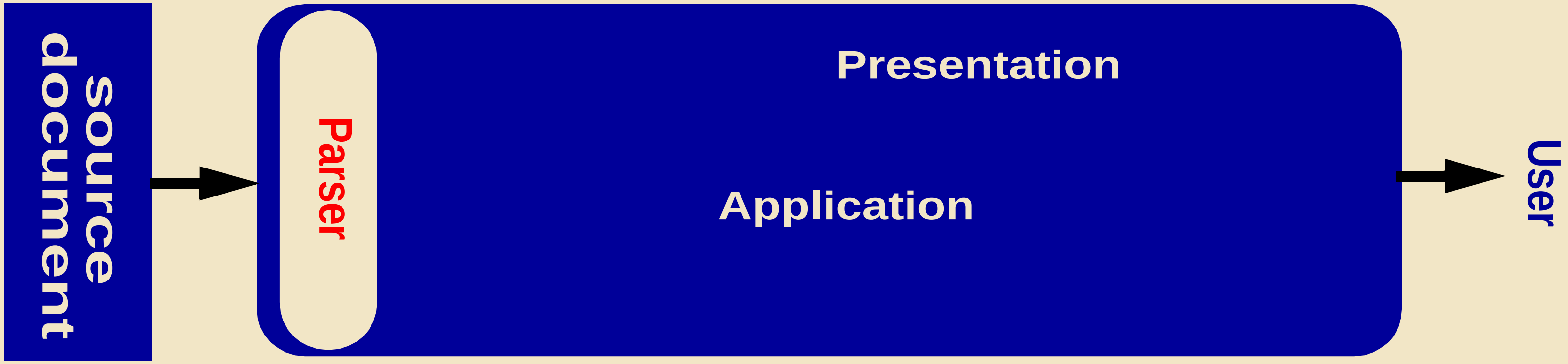
Domain specific

- Examples: product specific document standards (e.g. automobile and aircraft industry)
- Users need training, tailor-made tools might need to be developed
- Rich (domain-specific) semantics for further processing (validation, indexing, etc.)

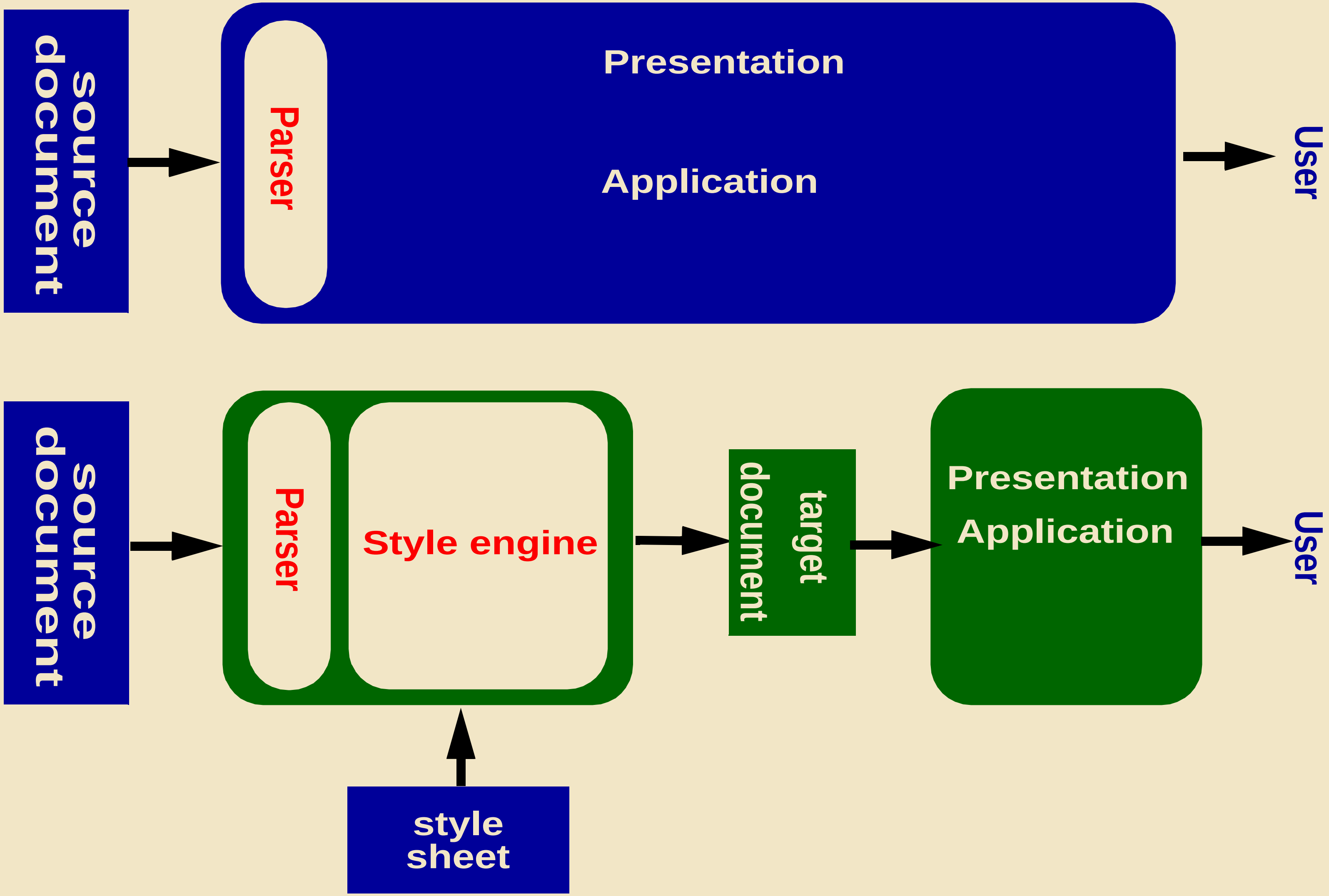
Consequence:

- Need tools tailored to domain-specific document formats or ...
- Generic tools that can process user-defined markup
  - no predefined (presentation) semantics

Presentation of domain dependent formats



Presentation of domain dependent formats



Beyond presentation semantics

Document-oriented semantics

- static: style and layout (e.g. style sheets, focus second half of this talk)
- dynamic: scheduling & animation
- interaction: linking & forms

Other semantics:

- do not describe the document, but the domain of the document’s content
- can still be related to document: annotations & meta data
- languages such as RDF(S), DAML+OIL, etc.
- more on this in Semantic Web talk next week

Structured Documents on the Web

Jacco van Ossenbruggen

Multimedia and Human-Computer Interaction Group
CWI: Centrum voor Wiskunde en Informatica
Amsterdam, The Netherlands

Presentation Outline:

- Introduction and historical background
- Multiple delivery publishing (MDP)
- MDP & the Web
- Style sheets
- Conclusion

Multiple delivery publishing on the Web

	W3C/HTML		
Markup	HTML		
Style	CSS		
Linking	<a href=		
Addressing	<a name=		

Multiple delivery publishing on the Web

	W3C/HTML		ISO/SGML
Markup	HTML		SGML
Style	CSS		DSSSL
Linking	<a href=		HyTime/TEI
Addressing	<a name=		HyTime/TEI

Multiple delivery publishing on the Web

	W3C/HTML	W3C/XML	ISO/SGML
Markup	HTML	XML	SGML
Style	CSS	CSS,XSLT,XSL	DSSSL
Linking	<a href=	XLink	HyTime,TEI
Addressing	<a name=	XPath,XPointer	HyTime,TEI

Structured Documents on the Web

Jacco van Ossenbruggen

Multimedia and Human-Computer Interaction Group
CWI: Centrum voor Wiskunde en Informatica
Amsterdam, The Netherlands

Presentation Outline:

- Introduction and historical background
- Multiple delivery publishing (MDP)
- MDP & the Web
- Style sheets
- Conclusion

Style sheets: HTML & CSS

HTML with embedded visual markup:

```
<h3 align="center">
  <font color="black">
     The Need for Style Sheets
 </font>
</h3>
```

versus HTML with separate CSS style sheet:

HTML:

```
<h3>The Need for Style Sheets</h3>
```

CSS:

```
h3 { text-align: center; color: black }
```

Style sheets: XML & CSS

Example fragment using MyOwnML (XML):

```
<product>
  <type>X112332</type>
  <color>dark blue</color>
  ...
</product>
```

With XML, your style sheet needs to specify more than just the style

CSS2:

```
product { display: list-item; ...}
type    { display: none;      ...}
color   { display: block;     ...}
```

Note:

- with XML, style sheets are no longer optional
- information presented with CSS remains in the same order
  - Source tree and target tree have similar structure (allows *cascading*)
- style properties are inherited via the source tree (!)

Transformations: XML and XSLT

What if the desired target tree differs radically from the source tree?

- assigning CSS properties will not suffice
- need a language to describe XML (tree) transformations:
  - XSL Transformations (XSLT)
  - more on XSL later!

XSLT

- Transforms from XML to:
  - XML (includes XHTML)
  - HTML (for legacy browsers, use old SGML syntax)
  - plain text (can be used to generate other text formats such as RTF, BibTeX)
- Uses XML syntax (unlike CSS)
  - so you can transform XSLT using XSLT
- Because the structure of the target tree and source tree can differ:
  - XSLT style sheets can be chained, not cascaded

XSLT template rules

Transformations are described as a set of one or more *template rules*

Each template rule consists of two parts:

- A pattern that is matched against the source tree: the *selector*
- A template to be filled in and added to the result tree

XSLT selectors are based on XPath, e.g:

```
/
*
product
color|type
product/color
catalog//product
text()
id("W11")
product[1]
@class
```

XSLT: Example (1)

A single template rule may already be sufficient...

```
<xsl:template match="/">
 <html><head>
   <title>Product Report Summary</title>
  </head><body>
  <p>...<table>
   <tr><td>
    <xsl:value-of select="product/type"/>
   </td><td>
    <xsl:value-of select="product/color"/>
  <td> ... </tr></table>...</body></html>
</xsl:template>
```

XSLT: Example (2)

... or a style sheet can contain many (smaller) template rules

```
<xsl:template match="/">
  <table>
   <xsl:apply-templates/>
  </table>
</xsl:template>

<xsl:template match=”product”>
  <tr>
   <xsl:apply-templates/>
  </tr>
</xsl:template>

<xsl:template match=”color|type”>
  <td>
   <xsl:apply-templates/>
  </td>
</xsl:template>

...
```

Style sheets: Formatting objects(1)

All these style sheet examples do actually *two* things:

- specify how an XML document should be presented
- specify how that presentation should be encoded in HTML

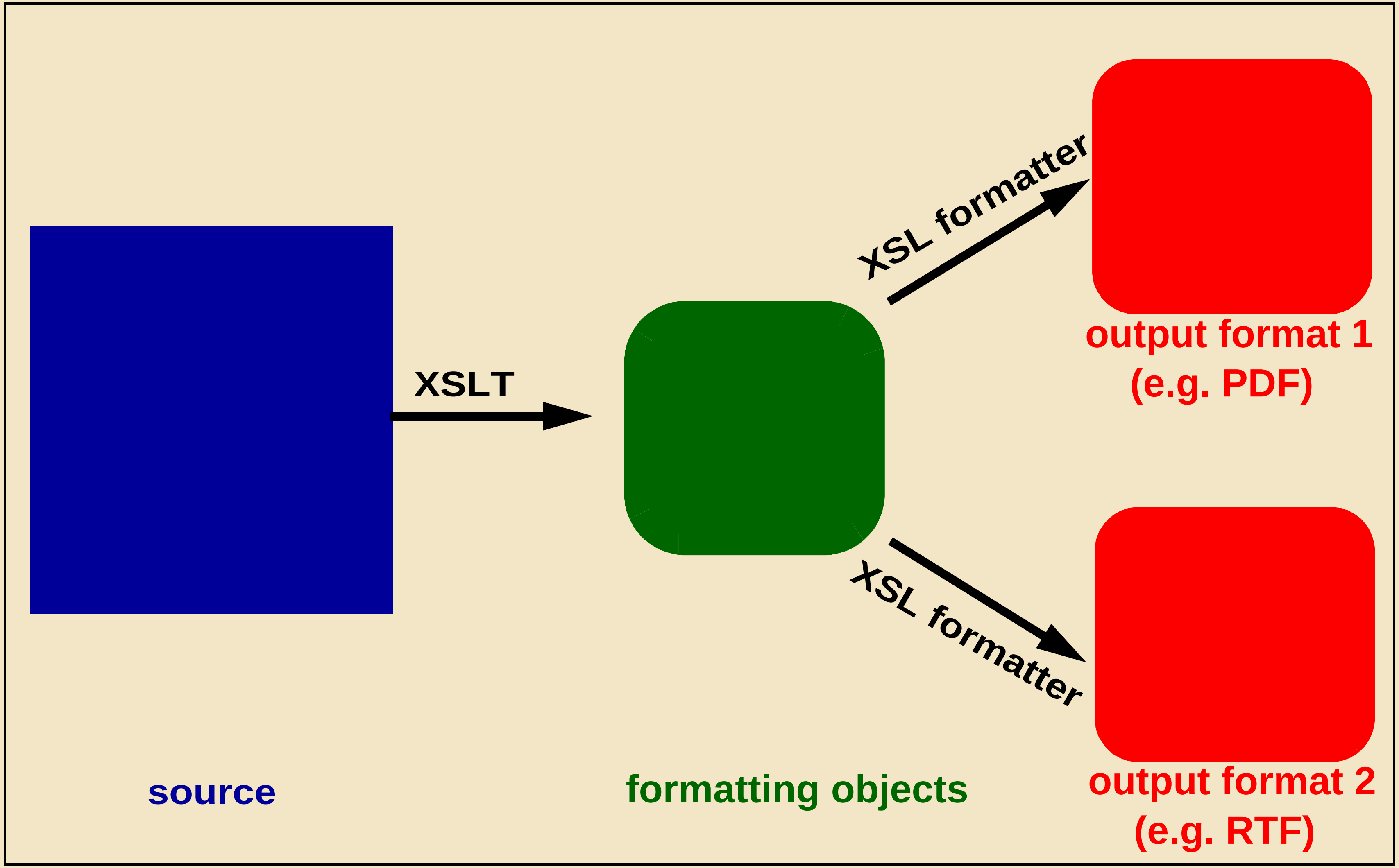
Drawbacks:

- need to start all over again for target formats other than HTML
- limited by the presentation capabilities of HTML & CSS

Solution:

- design new target language (argh!)
- a language that is designed to describe formatting semantics
- such a language is called a *formatting vocabulary*
- elements in the language are called *formatting objects* (FO)
- Example: the formatting vocabulary defined by XSL
    - fo:block, fo:flow, fo:footnote, fo:external-graphic, fo:page-sequence
- XSL suited for on-line and paper-based formatting

Style sheets: Formatting objects(2)



Style sheets: Formatting objects(3)

Advantages:

- Style sheets can be independent from final-form presentation format
- Formatting objects have more advanced formatting semantics than HTML/CSS

Disadvantages

- Yet another layer of abstraction
- Relative little tool support (XSL became a W3C Recommendation on 15 October 2001)
- XSL FOs are not suited for all output media (SMIL, SVG etc.)

Multiple delivery publishing wrap up

Advantages:

- Longevity
- Reusability
- Flexibility & Tailorability

Disadvantages:

- Complexity
- High dependency on tools (?!)
- Training
- High Initial investment

Works best for content-driven material

- becomes cheaper due to massive use on the Web
- free tool support
  - XML parsers, XSLT engines, XSL FO formatters, etc.
- many “off-the-shelf” source & target formats to choose from
  - XHTML, SVG, SMIL, MathML, Docbook, PDF, ...

Further reading

Overview pages at www.w3.org:

- http://www.w3.org/XML/
- http://www.w3.org/Style/XSL/
- http://www.w3.org/Style/CSS/

Recommendations (and drafts) at www.w3.org/TR/:

- http://www.w3.org/TR/xsl
- http://www.w3.org/TR/xslt
- http://www.w3.org/TR/REC-xml
- http://www.w3.org/TR/REC-CSS2

Tutorials and more

- http://www.xml.com
- http://www.mulberrytech.com/
- http://www.mulberrytech.com/quickref/ (personal favorite)