

# Creating XML

Hans C. Arents  
*senior IT market analyst*

**I.T. Works**  
*"Guiding the IT Professional"*

Innovation Center, Technologiepark 3, B-9052 Gent (Belgium), Tel: +32 (0)9 241 56 21 - Fax: +32 (0)9 241 56 56  
E-mail: [hca@itworks.be](mailto:hca@itworks.be) - Site: <http://www.itworks.be/> - Home: <http://www.arents.be/>

# Creating XML

- n Part 1: XML basics + XML DTDs 13u30 – 15u30  
*coffee break*
- n Part 2: **XML Schemas + tools overview** 16u00 – 17u30  
*questions & answers*

# XML Schemas + tools overview

Hans C. Arents  
*senior IT market analyst*

**I.T. Works**  
*"Guiding the IT Professional"*

Innovation Center, Technologiepark 3, B-9052 Gent (Belgium), Tel: +32 (0)9 241 56 21 - Fax: +32 (0)9 241 56 56  
E-mail: [hca@itworks.be](mailto:hca@itworks.be) - Site: <http://www.itworks.be/> - Home: <http://www.arents.be/>

# XML Schemas + tools overview

## n XML Schema basics

- how to define a schema for a document
- element and attribute types
- predefined datatypes

## n Tools overview

- DTD designers
- XML editors
  - for documents
  - for data
- XML parsers

# A sample XML document

```
<?xml version="1.0" encoding="utf-8"?>
<book isbn="0836217462">
  <title>Being a Dog Is a Full-Time Job</title>
  <author>Charles M. Schulz</author>
  <character>
    <name>Snoopy</name>
    <friend-of>Peppermint Patty</friend-of>
    <since>1950-10-04</since>
    <qualification>extroverted beagle</qualification>
  </character>
  <character>
    <name>Peppermint Patty</name>
    <since>1966-08-22</since>
    <qualification>bold, brash and tomboyish</qualification>
  </character>
</book>
```

# The DTD for this document

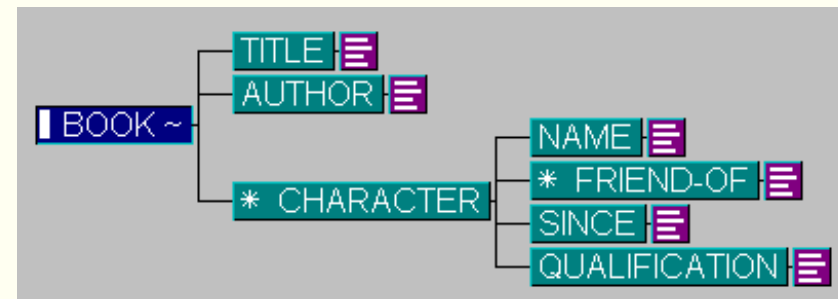
```
<!ELEMENT book          (title, author, character*)>
<!ATTLIST book          isbn CDATA #REQUIRED>

<!ELEMENT title         (#PCDATA)>

<!ELEMENT author        (#PCDATA)>

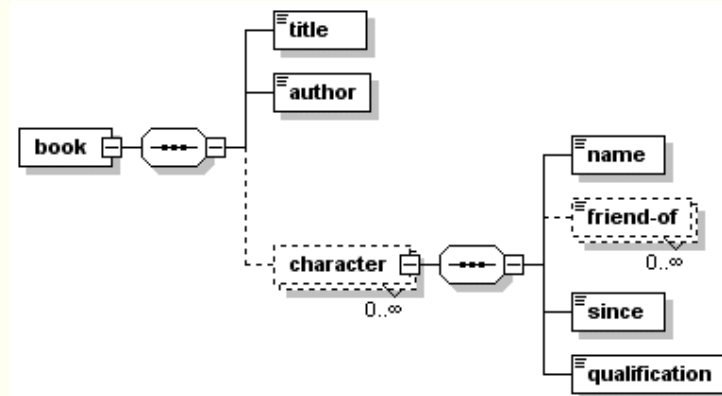
<!ELEMENT character     (name, friend-of*, since, qualification)>

<!ELEMENT name          (#PCDATA)>
<!ELEMENT friend-of    (#PCDATA)>
<!ELEMENT since         (#PCDATA)>
<!ELEMENT qualification (#PCDATA)>
```



# A first XML Schema for this document

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
  <xsd:element name="book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="title" type="xsd:string"/>
        <xsd:element name="author" type="xsd:string"/>
        <xsd:element name="character" minOccurs="0" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="name" type="xsd:string"/>
              <xsd:element name="friend-of" type="xsd:string"
                minOccurs="0" maxOccurs="unbounded"/>
              <xsd:element name="since" type="xsd:date"/>
              <xsd:element name="qualification" type="xsd:string"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
      <xsd:attribute name="isbn"
        type="xsd:string"
        use="required"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```



Generated with XMLSpy Schema Editor [www.xmlspy.com](http://www.xmlspy.com)

# Defining element and attribute types

**n** `xsd:element` declares an element and assigns it a type

**n** `xsd:attribute` declares an attribute and assigns it a type

**n** Every element or attribute is of a certain type

- complex type
- simple type

**n** Complex types can have child elements and attributes

- sequencing of child elements defined by a *compositor*
  - `sequence` DTD equivalent: `,`
  - `choice` DTD equivalent: `|`
  - `all` DTD equivalent: `&` (with restrictions)
- cardinality of child elements defined by an *occurrence* attribute
  - `minOccurs` (values = 0 ... unbounded, default value = 1)
  - `maxOccurs` (values = 0 ... unbounded, default value = 1)

**n** Simple types cannot have child elements or attributes



# Simple content element

n Simple content element = element without child elements

```
<xsd:element name="book">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="isbn" type="xsd:string" use="required"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

will validate the following element:

```
<book isbn="0836217462">
  Funny book by Charles M. Schulz.
  Its title (Being a Dog Is a Full-Time Job) says it all !
</book>
```

# Empty content element

n Empty content element = element without contents

```
<xsd:element name="book">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:restriction base="xsd:anyType">
        <xsd:attribute name="isbn" type="xsd:string" use="required"/>
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
```

or with the shorthand notation:

```
<xsd:element name="book">
  <xsd:complexType>
    <xsd:attribute name="isbn" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
```

will validate the following element:

```
<book isbn="0836217462"/>
```

# Mixed content element

n Mixed content element = element with child elements and text

```
<xsd:element name="book">
  <xsd:complexType mixed="true">
    <xsd:sequence>
      <xsd:element name="title" type="xsd:string"/>
      <xsd:element name="author" type="xsd:string"/>
    </xsd:sequence>
    <xsd:attribute name="isbn" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
```

will validate the following element:

```
<book isbn="0836217462">
  Its title (<title>Being a Dog Is a Full-Time Job</title>) says it all!
  Funny book by <author>Charles M. Schulz</author>.
</book>
```

will not validate the following element:

```
<book isbn="0836217462">
  Funny book by <author>Charles M. Schulz</author>.
  Its title (<title>Being a Dog Is a Full-Time Job</title>) says it all!
</book>
```

# A second XML Schema for this document

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
  <!-- definition of simple type elements -->
  <xsd:element name="title" type="xsd:string"/>
  <xsd:element name="author" type="xsd:string"/>
  <xsd:element name="name" type="xsd:string"/>
  <xsd:element name="friend-of" type="xsd:string"/>
  <xsd:element name="since" type="xsd:date"/>
  <xsd:element name="qualification" type="xsd:string"/>
  <!-- definition of attributes -->
  <xsd:attribute name="isbn" type="xsd:string"/>
  <!-- definition of complex type elements -->
  <xsd:element name="character"><xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="name"/>
      <xsd:element ref="friend-of" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="since"/><xsd:element ref="qualification"/>
    </xsd:sequence>
  </xsd:complexType></xsd:element>
  <xsd:element name="book"><xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="title"/><xsd:element ref="author"/>
      <xsd:element ref="character" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute ref="isbn" use="required"/></xsd:complexType>
  </xsd:element>
</xsd:schema>
```

# Predefined datatypes

## n Primitive datatypes

- `string`, `boolean`, `URI`, ...
- numeric types:
  - `integer`, `float`, `double`, ...
- time types:
  - `date`, `time`, `month`, `year`, `century`, ...

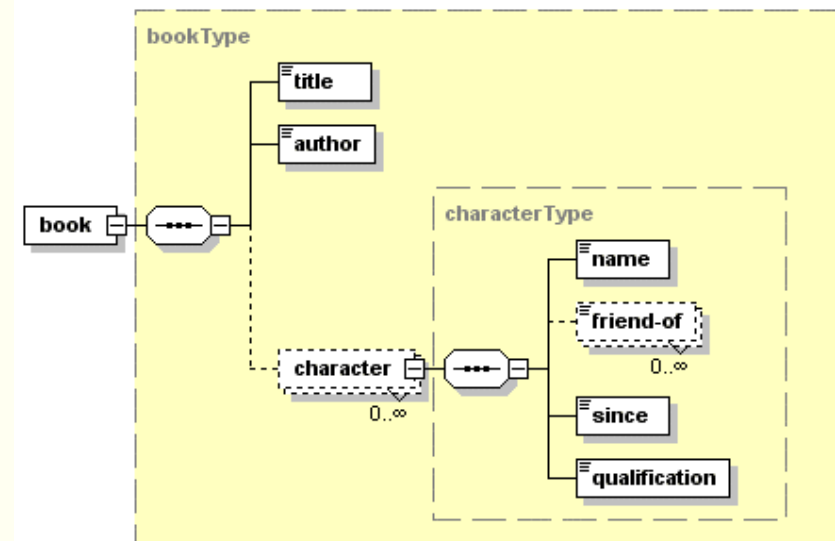
## n Possibility to create custom datatypes

- *basetype* restricted through the use of *facets*
  - e.g. for `string`:
    - `enumeration`, `pattern`
    - `length`, `minLength`, `maxLength`
  - e.g. for `decimal`:
    - `precision`, `scale`
    - `enumeration`, `pattern`
    - `minInclusive`, `maxInclusive`, `minExclusive`, `maxExclusive`

# A third XML Schema for this document

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
  <!-- definition of simple types -->
  <xsd:simpleType name="nameType">
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="32"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="sinceType">
    <xsd:restriction base="xsd:date"/>
  </xsd:simpleType>
  <xsd:simpleType name="descType">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>
  <xsd:simpleType name="isbnType">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[0-9]{10}"/>
    </xsd:restriction>
  </xsd:simpleType>
```

schema continues on next slide ...



Generated with XMLSpy Schema Editor [www.xmlspy.com](http://www.xmlspy.com)

# A third XML Schema for this document

... schema continues from next slide

```
<!-- definition of complex types -->
<xsd:complexType name="characterType">
  <xsd:sequence>
    <xsd:element name="name" type="nameType"/>
    <xsd:element name="friend-of" type="nameType"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="since" type="sinceType"/>
    <xsd:element name="qualification" type="descType"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="bookType">
  <xsd:sequence>
    <xsd:element name="title" type="nameType"/>
    <xsd:element name="author" type="nameType"/>
    <xsd:element name="character" type="characterType"
      minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="isbn" type="isbnType" use="required"/>
</xsd:complexType>
<!-- reference to "bookType" to define the "book" element -->
<xsd:element name="book" type="bookType"/>
</xsd:schema>
```

# Derivation of simple types: union and list

```
<xsd:simpleType name="isbnType">
  <xsd:union>
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:pattern value="[0-9]{10}" />
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType>
      <xsd:restriction base="xsd:NMTOKEN">
        <xsd:enumeration value="TBD" />
        <xsd:enumeration value="NA" />
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:union>
</xsd:simpleType>
```

valid ISBN number or "TBD" or "NA"

```
<xsd:simpleType name="isbnTypes">
  <xsd:list itemType="isbnType" />
</xsd:simpleType>
```

whitespace-separated list of ISBN numbers

```
<xsd:simpleType name="isbnTypes8">
  <xsd:restriction base="isbnTypes">
    <xsd:minLength value="1" />
    <xsd:maxLength value="8" />
  </xsd:restriction>
</xsd:simpleType>
```

list of 1 to 8 whitespace-separated ISBN numbers



# Groups of elements and attributes

```
<!-- definition of an element group -->
<xsd:group name="mainBookElements">
  <xsd:sequence>
    <xsd:element name="title" type="nameType"/>
    <xsd:element name="author" type="nameType"/>
  </xsd:sequence>
</xsd:group>

<!-- definition of an attribute group -->
<xsd:attributeGroup name="bookAttributes">
  <xsd:attribute name="isbn" type="isbnType" use="required"/>
  <xsd:attribute name="available" type="xsd:string"/>
</xsd:attributeGroup>
```

can be used in the definition of a complex type:

```
<xsd:complexType name="bookType">
  <xsd:sequence>
    <xsd:group ref="mainBookElements"/>
    <xsd:element name="character" type="characterType"
      minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="bookAttributes"/>
</xsd:complexType>
```

# Documenting an XML Schema

## n Alternative to XML comments and processing instructions

```
<xsd:element name="book">
  <xsd:annotation>
    <!-- human readable documentation -->
    <xsd:documentation>
      top level element of the book document
    </xsd:documentation>
    <!-- application specific information -->
    <xsd:appinfo source="http://www.somesite.com/someApp/">
      ... some application specific info ...
    </xsd:appinfo>
  </xsd:annotation>
  ...
</xsd:element>
```

## n XML Schema Adjunct Framework:

- for adding processing-specific information to XML Schemas
  - mappings to relational databases
  - business rules for additional validation
  - indexing parameters for native XML databases
- [http://www.extensibility.com/tibco/resources/saf\\_dec2000.htm](http://www.extensibility.com/tibco/resources/saf_dec2000.htm)

# XML Schema design guidelines

n Guidelines developed collectively by discussions among members of the xml-dev list group

<http://www.xml.org/xml-dev/>

n Purpose:

- describe the pros and cons of certain design choices
- enable a schema designer to make intelligent design decisions
- note: XML Schema specification offers no guidelines or best practices

n Latest version:

<http://www.xfront.com/BestPractices.html>

# DTD designers

## n Purpose:

- graphical tool to develop DTDs

## n Use:

- visualize the structure of an existing DTD
- adapt an existing DTD by dragging around elements
- create a new DTD by inserting and dragging around elements

## n Result:

- syntactically correct DTD syntax
- validated DTD which can be used for document/data markup

# Microstar *Near & Far Designer*

- n XML version of successful existing SGML DTD designer
  - ü easy to use graphical interface
  - ü conversion from SGML to XML DTDs
  - ü good validation and syntax error checking
  - ü comprehensive reporting on DTD structure
    - DTD structure maps for analysis and design reviews
    - element and attribute reports for SAX/DOM programming
  - ü comes with lots of useful example DTDs (DocBook, HTML, ...)
  - û focuses on *document* DTD design only
  - û only generates “classic” DTDs
  - û shows its SGML tool origins

n <http://www.microstar.com/>

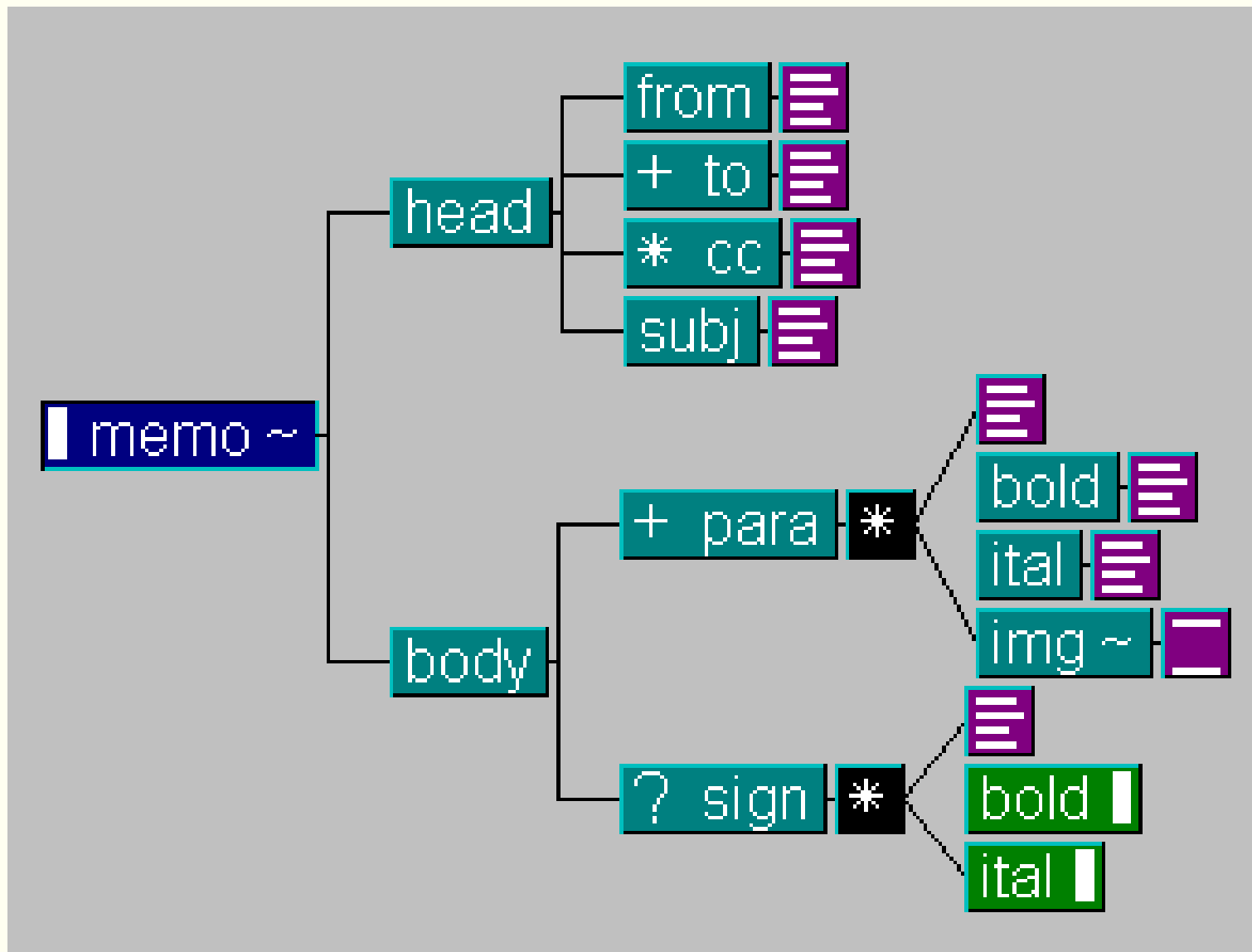
# An example DTD

```
<!-- DTD for simple memoranda                                -->
<!ENTITY % text "#PCDATA | bold | ital"                    >

<!--          NAME CONTENT MODEL                            -->
<!ELEMENT memo (head , body)                               >
<!ELEMENT head (from , to+ , cc* , subj)                   >
<!ELEMENT body (para+ , sign?)                             >
<!ELEMENT from (#PCDATA)                                    >
<!ELEMENT to    (#PCDATA)                                    >
<!ELEMENT cc    (#PCDATA)                                    >
<!ELEMENT subj  (#PCDATA)                                    >
<!ELEMENT para  (%text; | img)*                             >
<!ELEMENT sign  (%text;)*                                   >
<!ELEMENT bold  (#PCDATA)                                    >
<!ELEMENT ital  (#PCDATA)                                    >
<!ELEMENT img   EMPTY                                       >

<!--          ELEMENT NAME      VALUE                      DEFAULT      -->
<!ATTLIST memo      priority (Low | Medium | High) "Low"    >
<!ATTLIST img       src      ENTITY                      #REQUIRED
               type      CDATA                      #FIXED "GIF">
```

# Microstar *Near & Far Designer*



# Extensibility XML Authority

## n Newly developed XML document/data format designer

ü focuses on *document* and *data* format design

ü generates different XML schemas:

- “classic” DTDs, (a subset of) XML Schema, DCD, SOX, DDML, ...
- XML-Data (Reduced), import/export BizTalk compatible schemas

ü comes with extensive documentation on XML data modeling

- set of basic guidelines and best practices
- XMLschema.com site: schema validation/conversion

ü integrates with other tools:

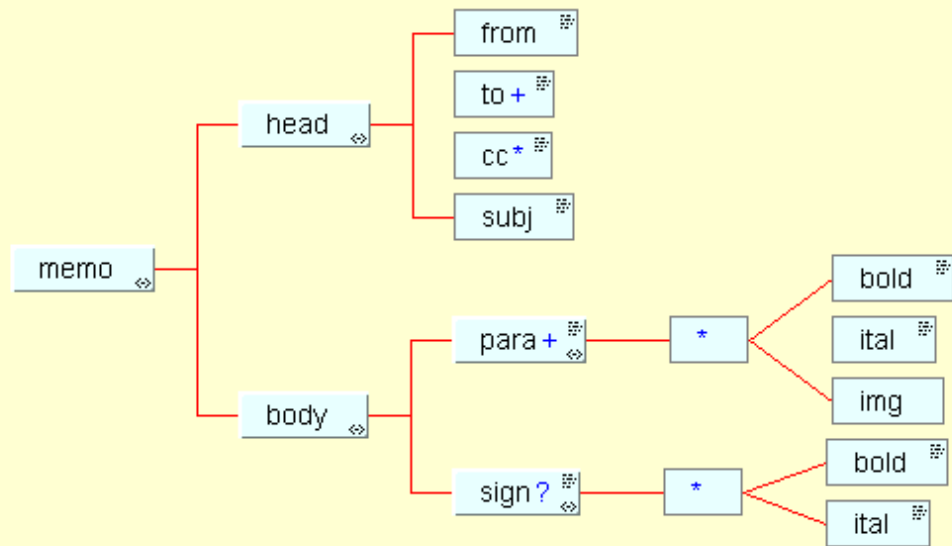
- part of *Turbo XML: XML Instance* (editor) + *XML Console* (management)
- can be used together with *XML Canon: XML schema repository*

û hard to use graphical interface

û shows it's written in Java

n <http://www.extensibility.com/>





memo is used by:

Element Type	Text	Elem.	Data	Content Model	Attributes
memo	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	(head , body)	priority
head	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	(from , to+ , cc* , subj)	Insert Attribute
body	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	(para+ , sign?)	
from	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
to	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
cc	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
subj	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
para	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	(%text   img)*	
sign	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	(%text)*	
bold	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
ital	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
img	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		src, type
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

# XML editors for documents

## n Purpose:

- assist in the creation and maintenance of document instances

## n Types of editor:

- XML-aware editor
- wordprocessor add-on

## n XML-aware editor

- use knowledge about DTD to guide the author when writing
  - SoftQuad *XMetaL*, ArborText *Epic*

## n Wordprocessor add-on

- map wordprocessor styles to document instance elements
  - i4i *S/4 TEXT*
  - Hypervision *WorX*
  - Corel *WordPerfect Office 2000*

# SoftQuad XMetaL

- n Based on Author/Editor SGML editor and HoTMetaL:
  - creates well-formed and valid XML documents (with tree view)
  - formatting based on CSS (internal) and XSL (external)
  - customization using VBScript, JavaScript, PerlScript, Python
  - COM-based architecture with support for DOM 1.0
- ü fair priced solution
- ü integrated with leading XML databases
- û built-in parser not fully compliant
- û faulty entity handling

n <http://www.xmetal.com/>

# SoftQuad XMetaL

The screenshot displays the SoftQuad XMetaL software interface, which is used for editing XML documents. The main window shows the XML source code for a document titled "Focus On Digital Cameras". The code includes sections for "Introduction" and "Our Picks", with various XML tags like <Section>, <Text>, <Table>, and <Image>.

Overlaid on the main window are several smaller windows and panels:

- Table of Contents:** A panel on the left side of the main window showing a hierarchical list of sections and subsections, including "Introduction" and "Our Picks".
- Preview Window:** A window titled "Focus On Digital Cameras" showing a rendered version of the XML content. It includes a title, an introduction paragraph, and a section titled "Our Picks" which features a small image of a camera and a list of recommendations.
- Table of Elements:** A panel on the right side of the main window showing a list of XML elements and their attributes, such as "id", "class", "style", "title", "lang", "dir", and "onclick".
- Table of Elements (Detailed):** A larger panel on the right side of the preview window showing a detailed list of elements and their attributes, including "onkeydown", "onkeyup", "abbr", "axis", "headers", "scope", and "rowspan".
- Table of Elements (Used):** A panel at the bottom right showing a list of elements that are currently used in the document, including "Section", "Caption", "Emphasis", "Figure", "Graphic", "Image", "ImageList", "Link", "LiteralLayout", "Note", "OrderedList", "Para", "ProgramListing", "Subscript", "Superscript", and "TABLE".

The interface also includes a menu bar (File, Edit, View, Insert, Tools, Table, Window, Help) and a toolbar with various icons for editing and navigation.

# ArborText *Epic*

## n Based on Adept Editor SGML editor:

- creates well-formed and valid XML documents (with tree view)
- formatting based on FOSI (proprietary) and XSL
- customization using ACL (Adept Command Language) only
- COM-based architecture with support for DOM 1.0

ü can handle large and complex documents

ü integrated with leading XML databases

û expensive solution

û hard to setup and adapt

n <http://www.arbortext.com/>

## i4i S/4 TEXT

**n** Based on patented "DataPipe" technology (map between proprietary internal application format and XML instance):

- creates valid XML documents
- formatting based on Word for Windows styles
- customization using expensive developers edition

**ü** works with Word for Windows

**ü** innovative underlying technology

**û** not user friendly

**û** hard to setup and adapt

**n** <http://www.i4i.com/>

# i4i S4/TEXT

The screenshot displays a Microsoft Word window titled "Microsoft Word - C:\Program Files\i4i\IPASAT\Sample Spec\TimExample1.xml". The document content includes:

[0009] It is desirable for the protective hockey equipment worn over the front of its position in a stable and secure manner for the duration that it is worn. comprised of vinyl, constructed in accordance with the instant invention i the same restriction of equipment slippage heretofore achieved by dispo tape applied in the same manner as that described as the preferred meth instant invention.

[0010] The use of vinyl material for this invention is responsible on its own, for restriction of equipment slippage identical in efficiency to that provided b athletic tape at a considerable reduction in expense, owing to the reusab

**What is Claimed is:**

[c1] A method of securing protective equipment to a hockey-player's leg by wrapping a r over the equipment and around the leg, said vinyl strap comprising: a rectangular stri gauge vinyl in lengths of 30" to 240", in widths of 1" to 1 1/2", in a solid color.

[c2] A method according to claim 1 wherein the vinyl strap is 120" to 240" and can be m shorter, customized-length strips.

The "New Specification" dialog box is open, showing a list of sections to include in the specification:

- Specification
- Title of Invention
- Copyright Statement
- Cross Reference to Related Applications
- Federal Research Statement
- Compact Disc Appendix
- Background of Invention
- Summary of Invention
- Brief Description of Drawings
- Detailed Description
- Program Listing Deposit
- Claims
- Abstract of Disclosure
- Figures

The "Claim Text" dialog box is also open, showing options:

- Insert New Claim
- Insert New Claim-text
- Insert Dependent Claim Reference
- OK

A drawing of a paperclip is shown in a small window, illustrating the invention.

# Hypervision WorX

## n Add-on to Word for Windows (specially developed WLL):

- creates valid XML documents
- formatting based on Word for Windows styles
  - real-time mapping of styles to XML elements

ü works with Word for Windows

ü tight integration

û not very intuitive to use

û expensive

n <http://www.hv1td.com/>



# Hypervision WorX

WorX SE Documentation.doc - Microsoft Word

File Edit View Insert Format Tools Table Window Help

Heading 1 Garamond 9 B I U

Documents Elements Structure

Technical Book

- Chapter
  - Title
  - para
  - Section (L1)
    - Title
    - para
  - Section (L1)
    - Title
    - para
  - Section (L1)
    - Title
    - para
  - Section (L1)
    - Title
    - para
  - Section (L1)
    - Title
    - para
  - Section (L1)
    - Title
    - para
  - variablelist
    - Entry
      - term
      - List Item
    - Entry
      - term
      - List Item
    - Entry
      - term
      - List Item
    - Entry
      - term
      - List Item

WorX Standard Edition (SE) v2.0

PREPARING XML DOCUMENTS USING MICROSOFT WORD

---

**BACKGROUND**

HyperVision Ltd. has developed WorX technology for Word, an authoring add-in designed for the preparation of XML documents using Microsoft Word. The initial version of WorX technology has demonstrated some of the concepts of XML authoring and the high fidelity model for creating XML from Microsoft Word (WorX 1.2). The next version of the product is to be called WorX Standard Edition (WorX SE) v 2.0. The reference to "Standard" is based on the expectation that it will be the choice of most organizations using XML in a "standard" manner (i.e., preparing XML documents compliant with a Schema). The use of the term "Standard" distinguishes it from other versions, in that it provides more control and more flexibility in creating XML from Word. Although there are some fundamental differences between the WorX 1.2 and the version being envisioned in WorX SE, experiences learned from the development of WorX 1.2 should help in the rapid development of WorX SE.

---

**THE PURPOSE OF THE PRODUCT**

WorX SE is intended for the creation of World Wide Web Consortium (W3C) Compliant XML v1.0 using Microsoft Word 2000. In contrast with most approaches where XML is created after the document is completed, WorX SE will exhibit, in real time, the structure of the document *as it is being prepared*. Therefore, the XML instance of the document can be obtained at any time. It is also intended to facilitate the reuse of information by allowing the incorporation of previously built XML documents, while keeping the structure compliant as much as possible, in the document being prepared. With WorX SE,

Page 1 Sec 1 1/4 At 2" Ln 3 Col 6 REC TRK EXT OVR

# Corel WordPerfect Office 2000

n Based on experience in SGML support for WordPerfect:

- creates valid XML documents
- formatting based on WordPerfect styles
- customization using Visual Basic for Applications

ü cheap solution

ü XML well integrated with wordprocessor

û not stable enough

n <http://www.corel.com/xml/>

# Corel WordPerfect Office 2000

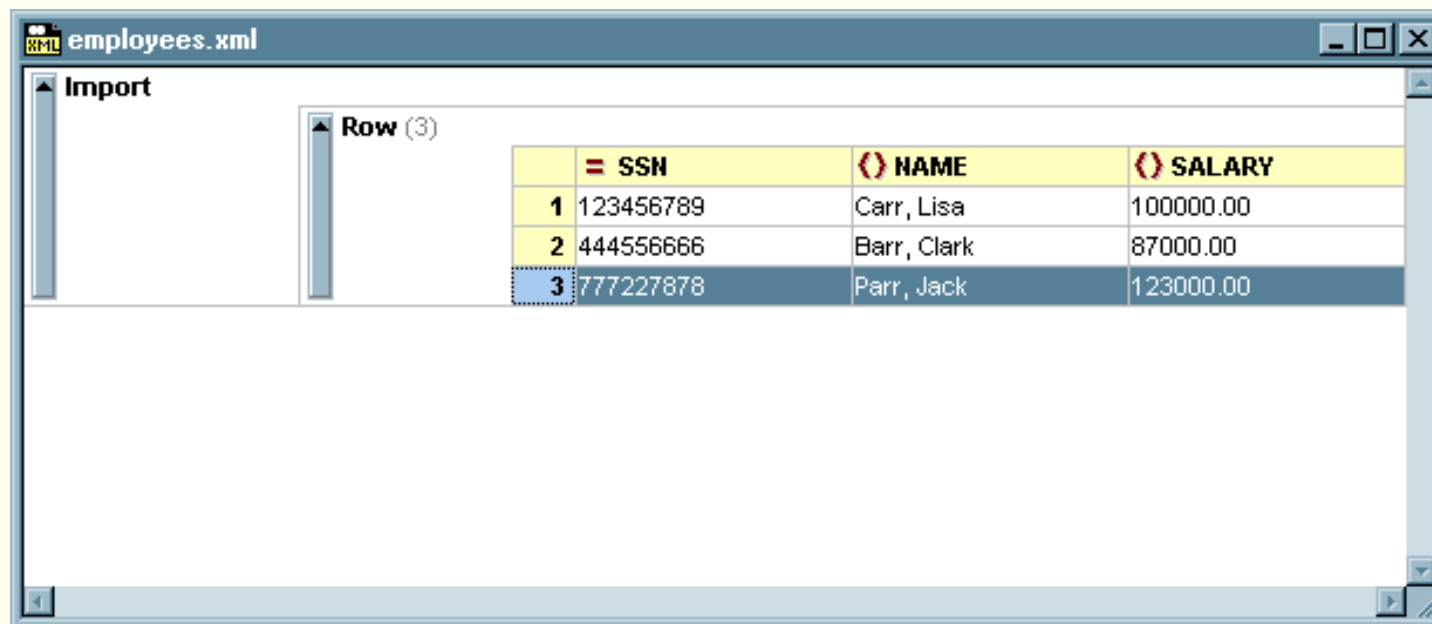
The image displays two overlapping windows from the Corel WordPerfect Office 2000 suite. The background window is 'WordPerfect XML Project Designer', which is used for defining XML templates. It features a 'Rule list' on the left with various HTML tags like 'a', 'address', 'b', 'br', 'center', 'code', 'em', 'h1', 'h2', 'h3', 'h4', 'h5', 'h6', 'p', 'pre', 'u', 'u+', 'u-', 'u+', 'u-', 'u+', 'u-'. The 'Hierarchy and Attributes' pane on the right shows a tree structure for an 'h1' tag with the attribute 'align = right'. Below this, there are fields for 'Start tag', 'End tag', 'Macro', and 'Selected Associations' (currently set to 'NORMAL').

The foreground window is 'WordPerfect 9 - C:\My Documents\wp.xml', showing a document in XML view. The document content is as follows:

```
[[[Title: WordPerfect 9 and XML]]]
[[[WordPerfect 9 and XML]]]
[[[Views from the trenches]]]
[[[By Greg Kohn and Michel Rodriguez]]]
]]
Introduction
[[[One of the most annoying problems with XML (and SGML before it) is the lack of editors/word processors that support it. Sure Emacs fits the bill, but would you let people familiar with traditional office suites or word processors use Emacs? The remaining options are a few pieces of Java software, [Adobe's] FrameMaker+SGML (at [$1499] a pop), or the even pricier Adept from [ArborText]. In other words, unless you have a healthy budget or an inclination to fiddle with all things technical and complex, XML and its cosmic potential are likely to pass right by. In a sense, the lack of XML editors defeats a primary purpose of markup languages such as XML: their cross-platform, cross-application functionality.]]]
[[[Corel's WordPerfect 9, which offers full, open XML/SGML support, stands to bring the power and promise of XML to the masses. Bundled with the WordPerfect Office 2000 suite, which also
```

# XML editors for data

- n Icon Information-Systems *XML Spy* <http://www.xmlspy.com/>
  - data-oriented XML editor (with XSL/XML Schema support)
  - can import a text file or database and convert it to XML data
  - can create a DTD or XML schema from well-formed XML data
  - helps you edit XML data which conforms to a DTD or XML schema
  - has a “database view” for displaying sequences of repeating elements



	SSN	NAME	SALARY
1	123456789	Carr, Lisa	100000.00
2	444556666	Barr, Clark	87000.00
3	777227878	Parr, Jack	123000.00

# Other XML editors

n Stilo *XMLDeveloper*

– <http://www.stilo.com/>

n XMLWriter

– <http://www.xmlwriter.net/>

n XML Pro

– <http://www.vervet.com/xmlpro.html>

n Xeena

– <http://www.alphaworks.ibm.com/tech/xeena>

n XML Editor Maker

– <http://www.alphaworks.ibm.com/tech/xmleditormaker>

n ...

# XML parsers

## n Purpose:

- parse the contents of an XML document

## n Types of parser:

- *well-formedness parser*: checks well-formedness
- *validating parser*: finds and reports markup errors
  - note: there exists no conformance testing of XML parsers

## n Stand-alone validating parsers

- validate and produce information for *external* processing
  - James Clark *EXPAT* (W) and *SP* (V)
  - IBM *XML4J* (V) and *XML4C* (V)
  - Sun *Java Project X* (V)

## n Validating parsers which are part of a convertor

- validate and produce information for *internal* processing
  - OmniMark Technologies *OmniMark*

# Validating XML parser

## n Java:

- IBM *XML4J*: fastest, Sun *Java Project X*: most compliant, ...

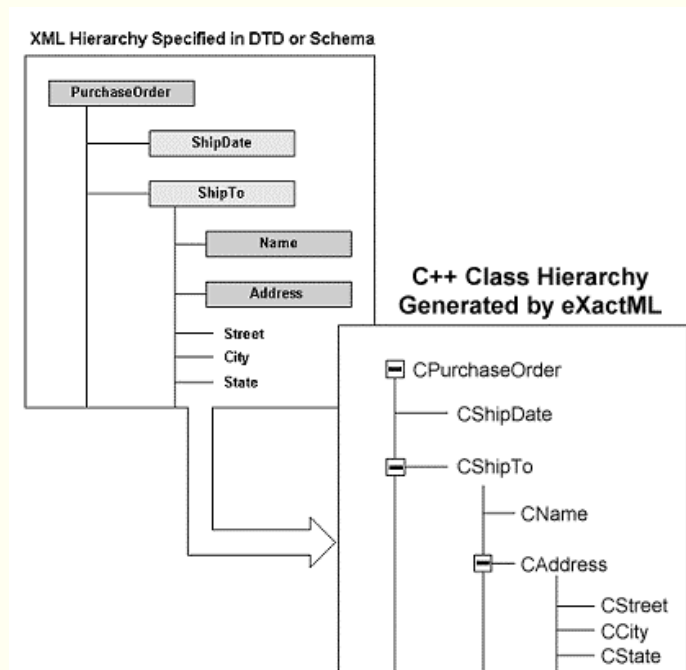
## n Other languages:

- IBM *XML4C* (C++), *XML::Parser* (Perl), ...

## n Parser generator:

- turn DTD or XML Schema into Visual Basic, Java or C++ classes
  - marshalling: class à XML
  - unmarshalling: XML à class
- check well-formedness and validity
- use classes directly instead of SAX/DOM
  - e.g. *eXactML*

<http://www.bristol.com/>



# Validating XML parser

## n Windows:

- Microsoft *MSXML* (IE 5.01 version) to be used in ASP, VB, ...
  - warning: supports pre-standard version of XSL (“MS-XSL”)
- Microsoft *MSXML Version 3.0* (October 2000)
  - supports XDR schemas, not XML Schemas
  - completely supports standard XSLT and XPath
  - user-defined extension functions in VBScript or JScript
  - optimizations for improved document throughput (2/3x faster)
    - e.g. server-side XSL stylesheet caching
  - can be installed alongside or as a replacement of the original *MSXML*
  - COM object accessible from C++, Visual Basic and scripting environments

n <http://www.netcrucible.com/xslt/msxml-faq.htm>