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 - Site: http://www.itworks.be/ - Home: http://www.arents.be/

Creating XML

- Part 1: XML basics + XML DTDs 13u30 – 15u30

  coffee break

- 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 - Site: http://www.itworks.be/ - Home: http://www.arents.be/

XML Schemas + tools overview

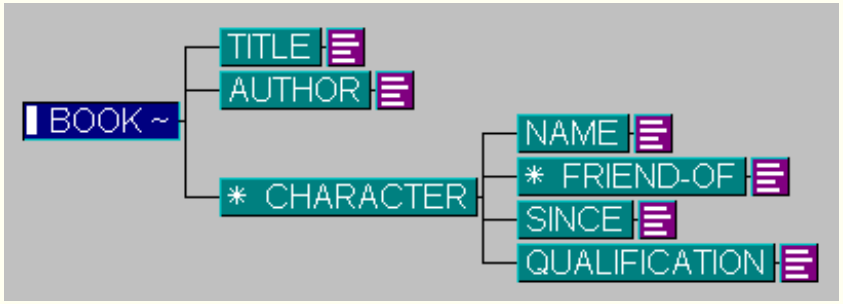
- XML Schema basics
  - how to define a schema for a document
  - element and attribute types
  - predefined datatypes
- Tools overview
  - DTD designers
  - XML editors
    - for documents
    - for data
  - XML parsers

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>
```

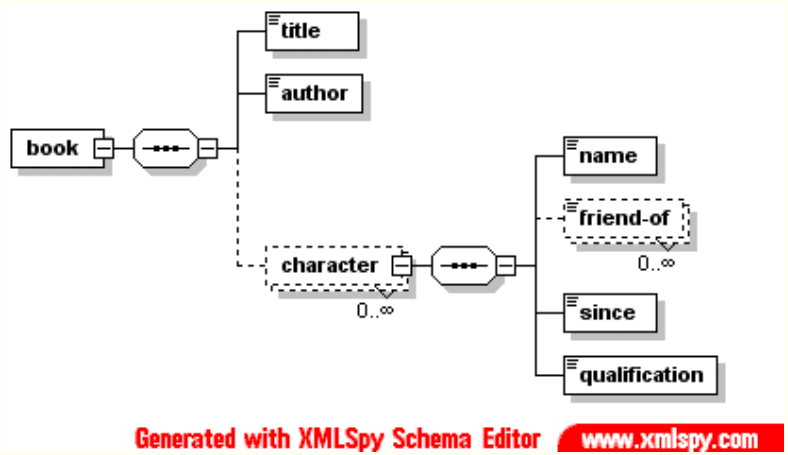
The DTD for this document

```
<!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>
```



Defining element and attribute types

- `xsd:element` declares an element and assigns it a type
- `xsd:attribute` declares an attribute and assigns it a type
- Every element or attribute is of a certain type
  - complex type
  - simple type
- 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`)
- Simple types cannot have child elements or attributes

Simple content element

- 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

- 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

- 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>
```

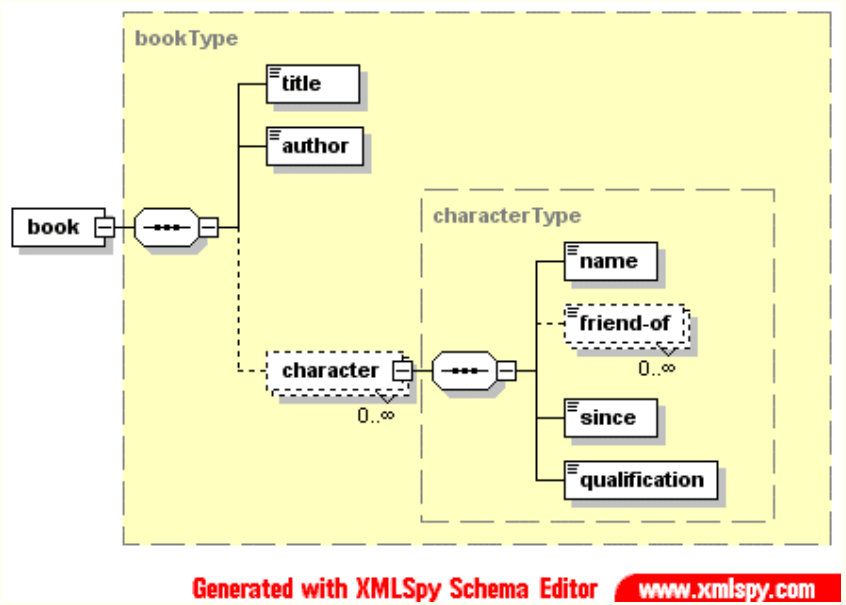
Predefined datatypes

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



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

- 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>
```

- 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

XML Schema design guidelines

- Guidelines developed collectively by discussions among members of the xml-dev list group
  http://www.xml.org/xml-dev/
- 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
- Latest version:
  http://www.xfront.com/BestPractices.html

DTD designers

- Purpose:
  - graphical tool to develop DTDs
- 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
- Result:
  - syntactically correct DTD syntax
  - validated DTD which can be used for document/data markup

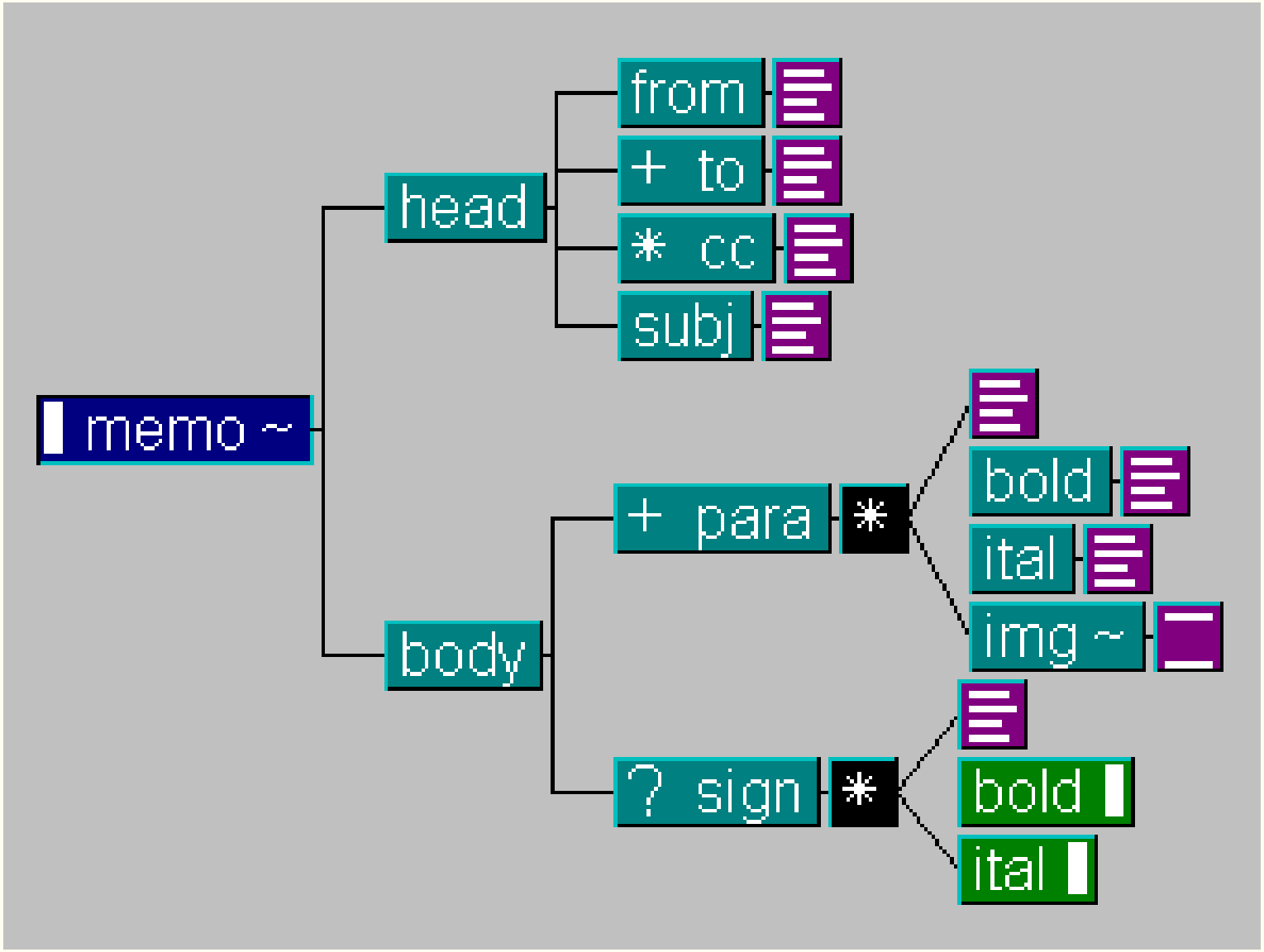
Microstar *Near & Far Designer*

- 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 it SGML tool origins
- **http://www.microstar.com/**

An example DTD

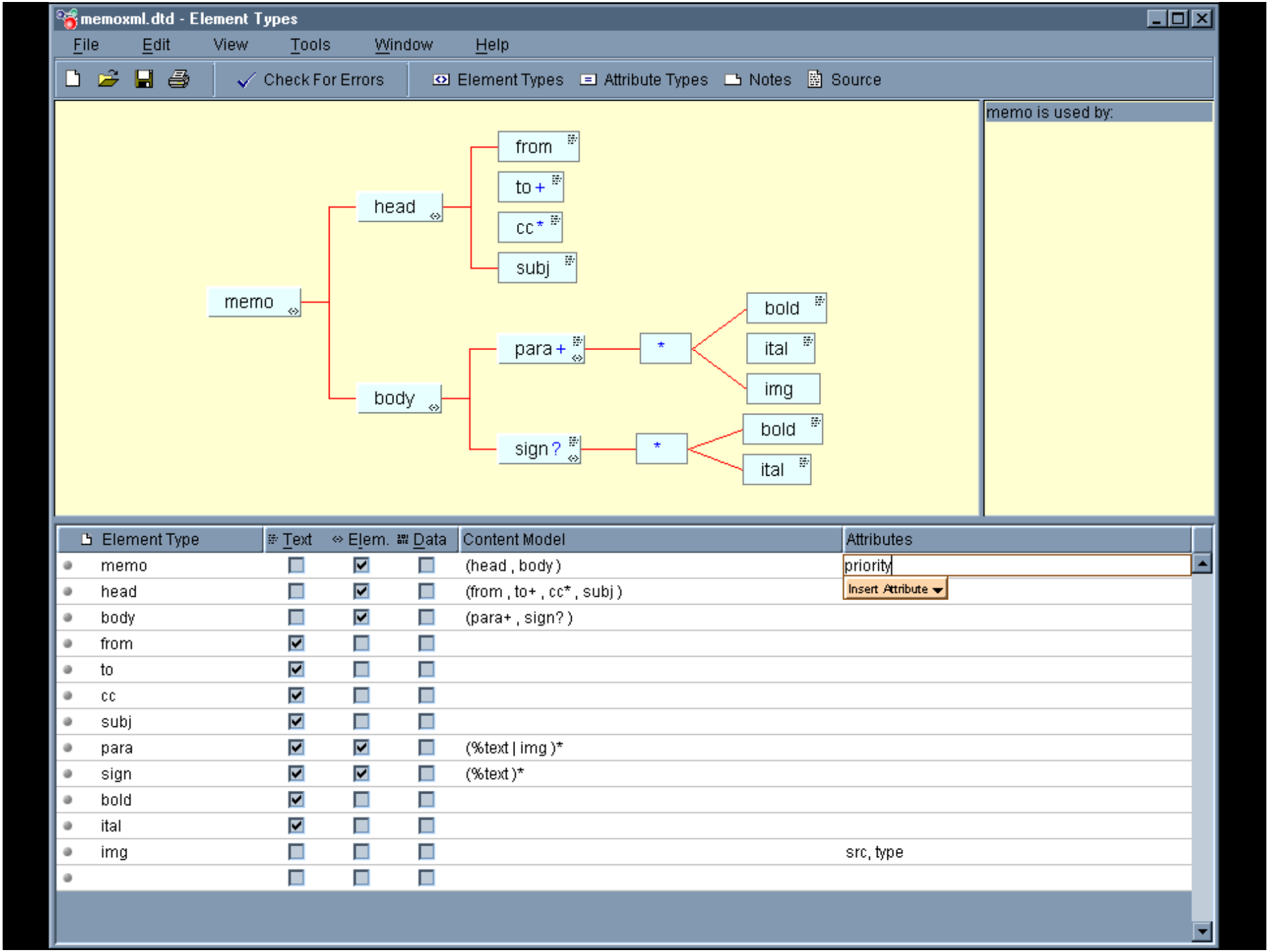
```
<!-- DTD for simple memoranda                                    -->
<!ENTITY % text "#PCDATA | bold | ital"                           >

<!--      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*

- 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
- **http://www.extensibility.com/**


memoxml.dtd - Element Types
File
Edit
View
Tools
Window
Help
Check For Errors
Element Types
Attribute Types
Notes
Source
memo is used by:
memo
head
from
to +
cc *
subj
body
para +
*
bold
ital
img
sign ?
*
bold
ital
Element Type
Text
Elem.
Data
Content Model
Attributes
memo (head , body)
priority
Insert Attribute
head (from , to+ , cc* , subj)
body (para+ , sign?)
from
to
cc
subj
para (%text | img)*
sign (%text)*
bold
ital
img
src, type

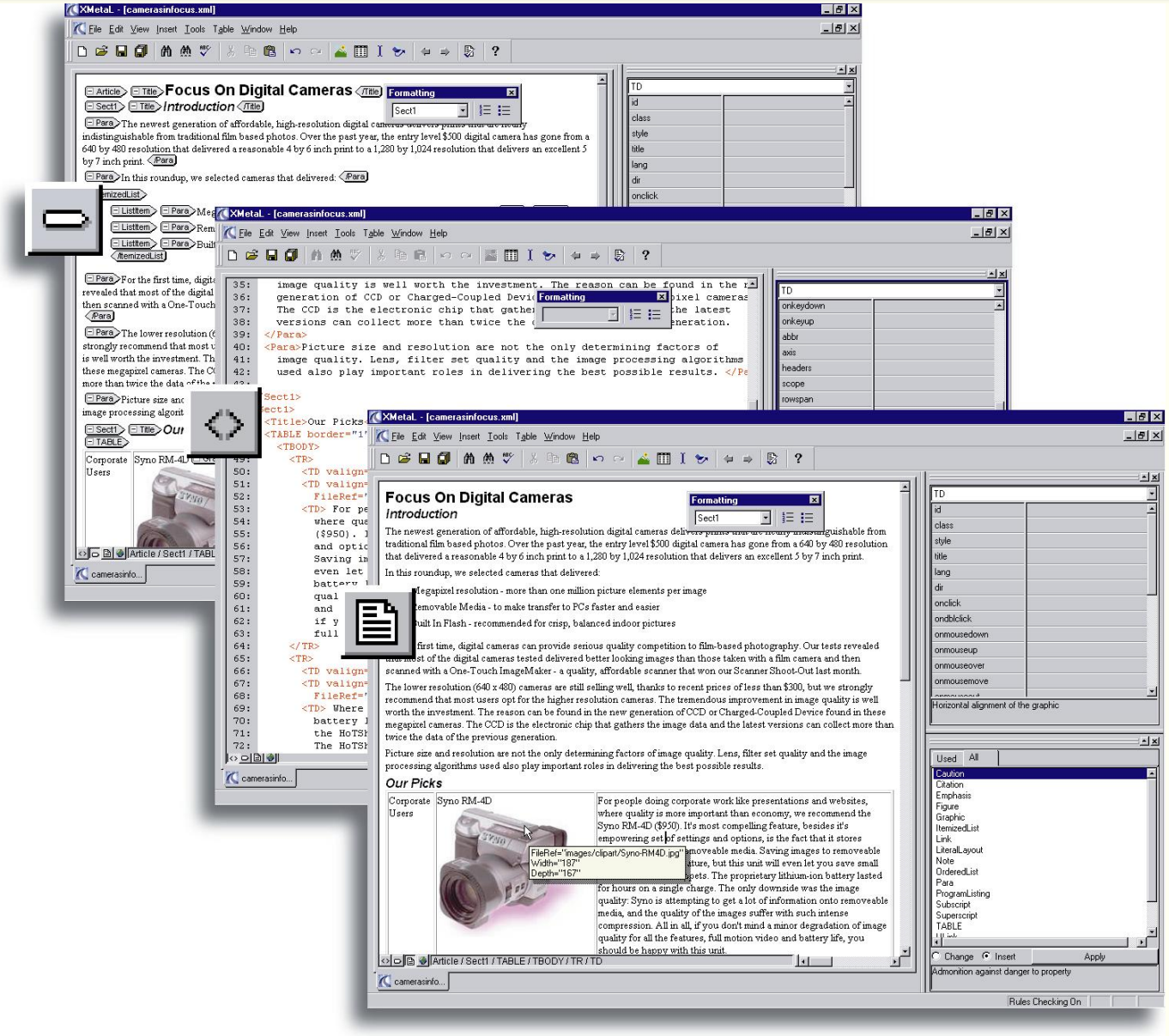
XML editors for documents

- Purpose:
  - assist in the creation and maintenance of document instances
- Types of editor:
  - XML-aware editor
  - wordprocessor add-on
- XML-aware editor
  - use knowledge about DTD to guide the author when writing
    - SoftQuad *XMetaL*, ArborText *Epic*
- Wordprocessor add-on
  - map wordprocessor styles to document instance elements
    - i4i *S/4 TEXT*
    - Hypervision *WorX*
    - Corel *WordPerfect Office 2000*

SoftQuad *XMetaL*

- 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
- **http://www.xmetal.com/**

SoftQuad *XMetaL*



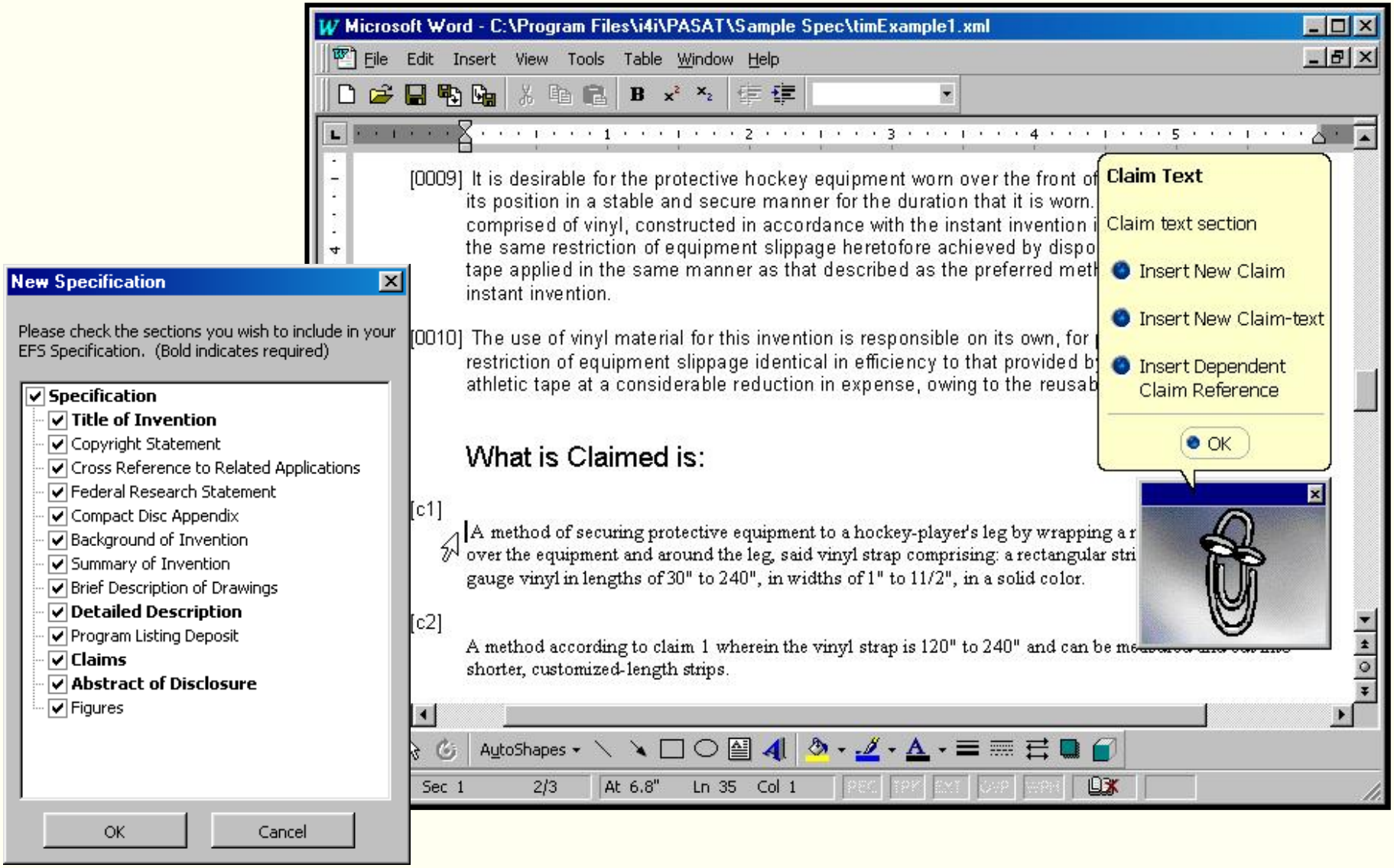
ArborText *Epic*

- 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
- **http://www.arbortext.com/**

i4i *S/4 TEXT*

- 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
- **http://www.i4i.com/**

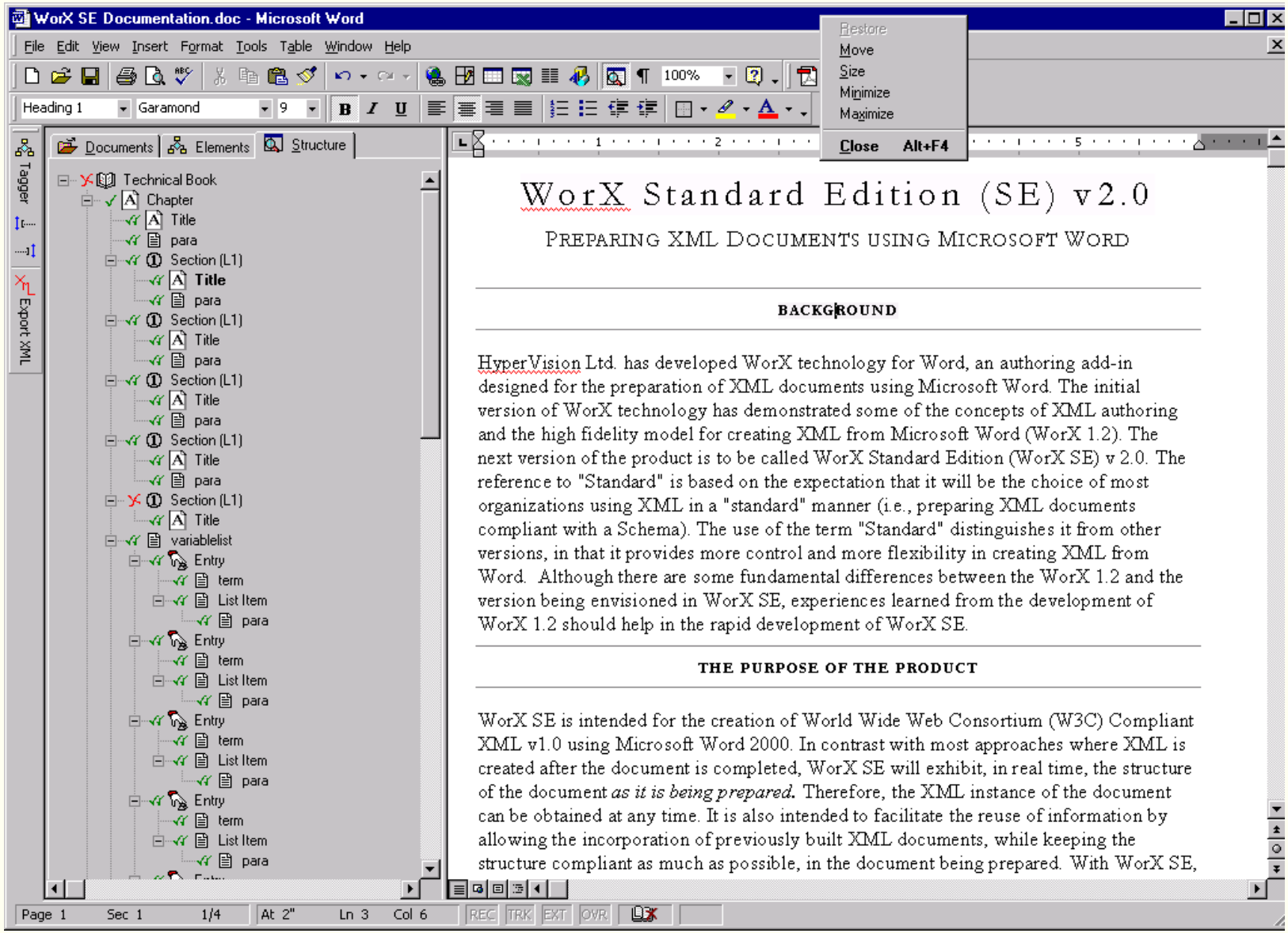
i4i *S4/TEXT*


Microsoft Word - C:\Program Files\i4i\PASAT\Sample Spec\timExample1.xml
File Edit Insert View Tools Table Window Help
[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 11/2", in a solid color.
[c2]
A method according to claim 1 wherein the vinyl strap is 120" to 240" and can be
shorter, customized-length strips.
AutoShapes
Sec 1 2/3 At 6.8" Ln 35 Col 1
Claim Text
Claim text section
Insert New Claim
Insert New Claim-text
Insert Dependent Claim Reference
OK
New Specification
Please check the sections you wish to include in your EFS Specification. (Bold indicates required)
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
OK
Cancel

Hypervision *WorX*

- 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
- **http://www.hvltd.com/**

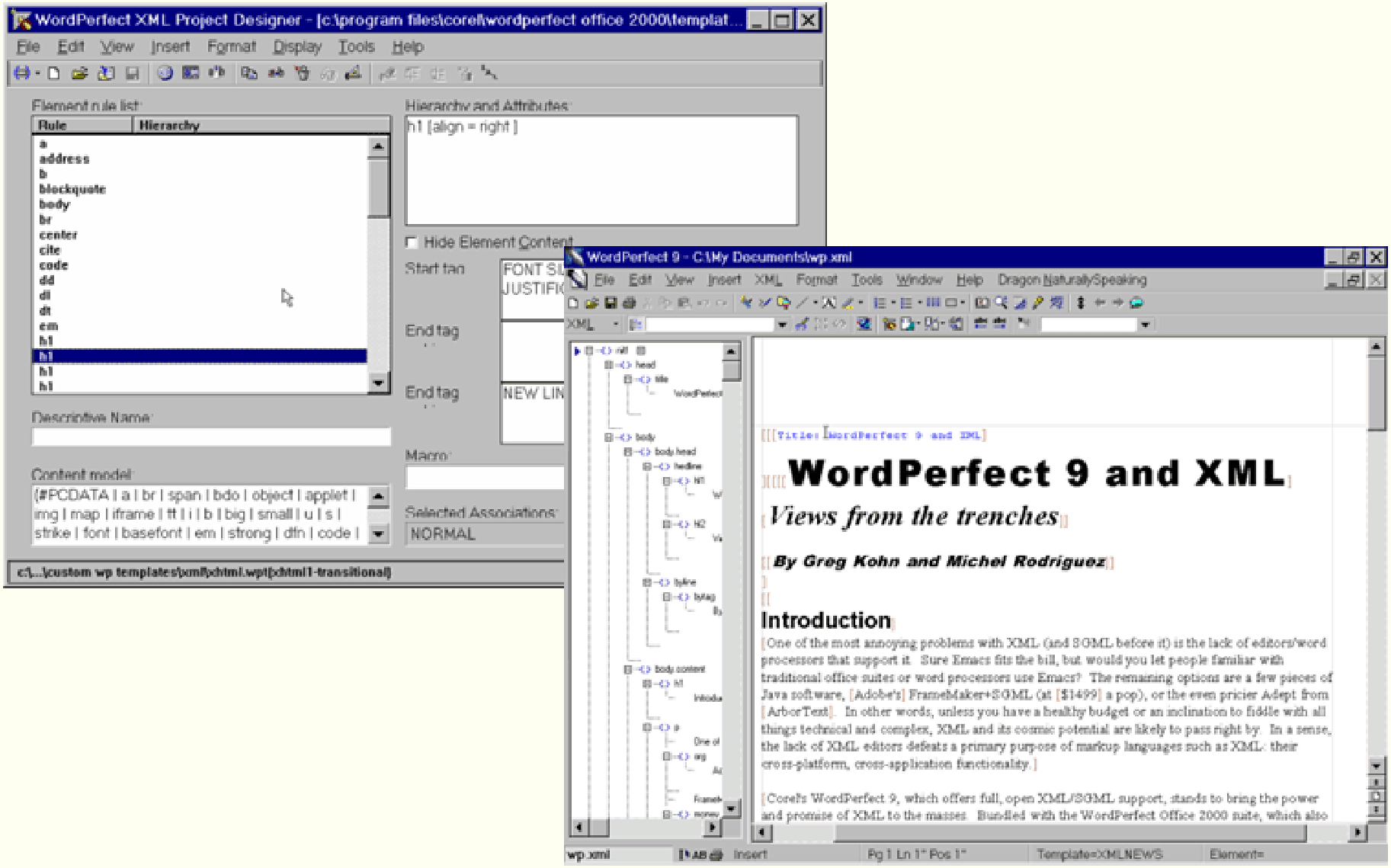
Hypervision *WorX*



Corel *WordPerfect Office 2000*

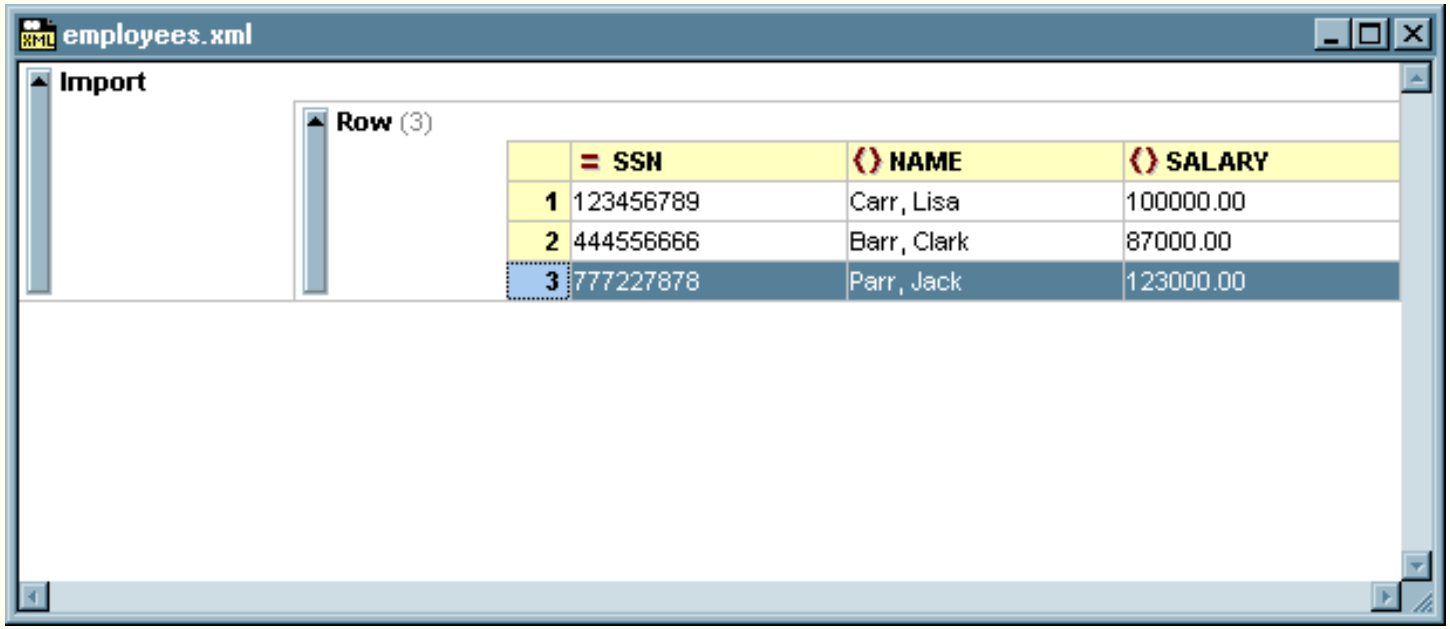
- Based on experience in SGML support for WordPerfect:
  - creates valid XML documents
  - formatting based on WordPerfect styles
  - customization using Visual Basic for Applications
  - ü cheap solution
  - ü XML well integrated with wordprocessor
  - û not stable enough
- **http://www.corel.com/xml/**

Corel *WordPerfect Office 2000*



XML editors for data

- 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



Other XML editors

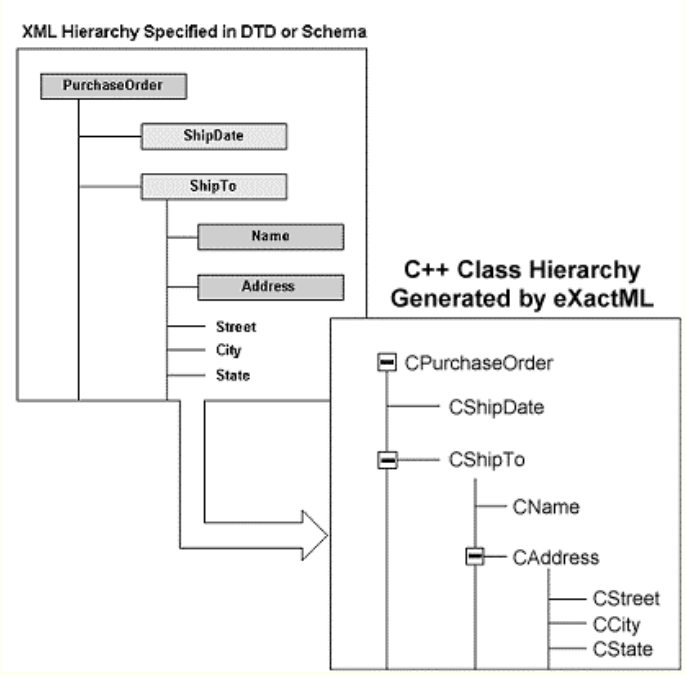
- Stilo *XMLDeveloper*
  - http://www.stilo.com/
- XMLWriter
  - http://www.xmlwriter.net/
- XML Pro
  - http://www.vervet.com/xmlpro.html
- Xeena
  - http://www.alphaworks.ibm.com/tech/xeena
- XML Editor Maker
  - http://www.alphaworks.ibm.com/tech/xmleditormaker
- …

XML parsers

- Purpose:
  - parse the contents of an XML document
- 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
- 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)
- Validating parsers which are part of a convertor
  - validate and produce information for *internal* processing
    - OmniMark Technologies *OmniMark*

Validating XML parser

- Java:
  - IBM *XML4J*: fastest, Sun *Java Project X*: most compliant, …
- Other languages:
  - IBM *XML4C* (C++), *XML::Parser* (Perl), …
- 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

- 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
- **http://www.netcrucible.com/xslt/msxml-faq.htm**