Introduction to XML DTD creation

Code: xml-dtd

Author and version

- Daniel K. Schneider
- Email: Daniel.Schneider@tecfa.unige.ch

- Version: 0.93 (modified 5/11/10 by DKS)

Prerequisites

- HTML
- Editing XML (being able to use a simple DTD)

Availability

url: **http://tecfa.unige.ch/guides/te/files/xml-dtd.pdf**

Objectives

- Being able to create a moderately complex XML DTD

Disclaimer

- There may be typos (sorry) and mistakes (sorry again)
- Please also consult a textbook !

1. Table of contents

2. Introduction - recall of previous knowledge

2.1 Structure and role of DTDs (Document Type Definitions)

DTD grammars are just a set of rules that define:

- a set of elements (tags) and their attributes that can be used;
- how elements can be embedded;
- different sorts of entities (reusable fragments, special characters).
- DTDs can’t define what the character data (element contents) and most attribute values look like.

Specification of a markup language

- The most important part is usually the DTD, but in addition other constraints can be added !
- The DTD does not identify the root element !
  - you have to tell the users what elements can be root elements
- Since DTDs can’t express data constraints, you may write out additional ones in a specification document
  - e.g. "the value of length attribute is a string composed of a number plus one of "cm", "inch", "em"

```
<size length="10cm">
```

2.2 Using a DTD with an XML document

4 ways of using a DTD

1. No DTD (XML document will just be well-formed)
2. DTD rules are defined inside the XML document
   - We get a "standalone" document (the XML document is self-sufficient)
   - DTD rules are inserted between brackets [...]

```
<!DOCTYPE hello [
   <!ELEMENT hello (#PCDATA)>
                  ]>
```

3. "Private/System" DTDs, the DTD is located on the system (own computer or the Internet)
   - ... that's what **you** are going to use when you write your own DTDs
   - DTD is identified by the URL after the "**SYSTEM**" keyword

```
<!DOCTYPE hello SYSTEM "hello.dtd">
```

4. Public DTDs, we use a name for the DTD.
   - means that both your XML editor and user software know the DTD
   - strategy used for common Web DTDs like XHTML, SVG, MathML, etc.
   - after the "**PUBLIC**" keyword you have to specify an official name and a backup URL that a validator could use.

```
<!DOCTYPE rss PUBLIC "-//Netscape Communications//DTD RSS 0.91//EN"
 "http://my.netscape.com/publish/formats/rss-0.91.dtd">
```

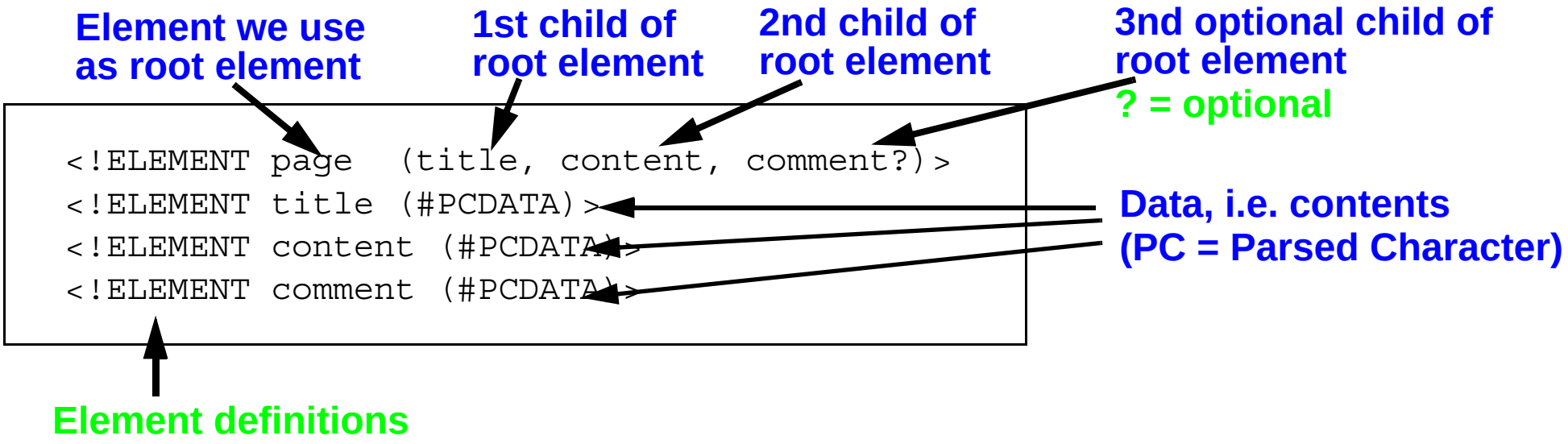
2.3 Recall of a simple example

Example 2-1: Hello text with XML

- root is **<page>**

```
<page>
 <title>Hello friend</title>
 <content>
      Here is some content :)
 </content>
 <comment>
      Written by DKS/Tecfa, adapted from S.M./the Cocoon samples
 </comment>
</page>
```

A DTD that would validate the document



3. Defining elements

Syntax of a DTD rule to define elements:

```
Syntax: <!ELEMENT tag_name child_element_specification>
```

Child_element_specification may contain:

- A combination of child elements according to combination rules

```
<!ELEMENT page  (title, content, comment?)>
```

- Mixed contents, i.e. child elements plus #PCDATA or ANY

```
<!ELEMENT para (strong | #PCDATA )*>
```

- #PCDATA (Just data)

```
<!ELEMENT title (#PCDATA)>
```

- ANY (only used during development)

```
<!ELEMENT para (ANY)*>
```

- EMPTY (the element has no contents)

```
<!ELEMENT person EMPTY>
```

Tag names

- Each tag name must start with a letter or an underscore ('_')
  followed by letters, numbers or the following characters: '_' , '-', '.', ':'

  Examples of illegal elements names:

```
<!ELEMENT 1st ...>
<!ELEMENT My Home ...>
```

3.1 Combination rules

A and B = tags	Explanation	DTD example	XML example

A , B	A followed by B	`<!ELEMENT person (name ,email?)>`	`<person>`
			`  <name>Joe</name>`
			`  <email>x@x.x</email>`
			`</person>`

A?	A is optional, (it can be present or absent)	`<!ELEMENT person (name, email?)>`	`<person>` `  <name>Joe</name></person>`

A+	At least one A	`<!ELEMENT person (name, email+)>`	`<person> <name>Joe</name>`
			`  <email>x@x.x</email></person>`
			`<person> <name>Joe</name>`
			`  <email>x@x.x</email>`
			`  <email>x@y.x</email>`
			`</person>`

A*	Zero, one or several A	`<!ELEMENT person (name, email*)>`	`<person>`
			`  <name>Joe</name>`
			`</person>`

A | B	Either A or B	`<!ELEMENT person (email | fax)>`	`<person> <name>Joe</name>`
			`  <email>x@x.x</email></person>`
			`<person> <name>Joe</name>`
			`  <fax>123456789</fax></person>`

(A, B)	Parenthesis will group and you can apply the above combination rules to the whole group	`<!ELEMENT list (name, email)+ >`	`<list>`
			`  <person> <name>Joe</name>`
			`  <email>x@x.x</email></person>`
			`</list>`

3.2 Special contents

Special elements	Explanation	DTD examples	XML example
#PCDATA	"Parsed Character Data" Text contents of an element. It should not contain any <,>,& etc.	`<!ELEMENT email (#PCDATA)>`	`<email>Daniel.Schneider @tecfa.unige.ch</email>`
ANY	Allows any non-specified child elements and parsed character data (avoid this !!!)	`<!ELEMENT person ANY>`	`<person> <c>text</c> <a>some <b>bbb</b> inside </a> </person>`
EMPTY	No contents	`<!ELEMENT br EMTPY>`	` `

Note about Mixed Content

- Mixed element contents contain both text and tags, usually in random order. Example:

```
<para> here is <a href="xx">link</a>. <b>Check</b> it out </para>
```

- You have to use the "|" construct for these
  - Good examples:

```
<!ELEMENT para (#PCDATA|a|ul|b|i|em)*>
<!ELEMENT p (#PCDATA | a | abbr | acronym | br | cite | code | dfn | em | img | kbd |
                     q | samp | span | strong | var )* >
<!ELEMENT p (#PCDATA | %font; | %phrase; | %special; | %form;)* >
```

  - Bad examples:

```
<!ELEMENT p (name, first_name, #PCDATA)*>
<!ELEMENT p ( (#PCDATA) |a|ul|b|i|em)*>
```

Example 3-1: A simple story grammar

url: http://tecfa.unige.ch/guides/xml/examples/xsd-examples/story-grammar.dtd

The THREADS element must contain one or more EPISODE elements
+ = at least one

All elements of STORY must be present in this order

Comment

Choose one
| = or

Choose as many as you like in random order

These elements only contain text

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!-- DTD to write simple stories -  VERSION 1.0 1/2007
     Made by Daniel K. Schneider / TECFA / University of Geneva -->

<!ELEMENT STORY (title, context, problem, goal, THREADS, moral, INFOS)>
<!ATTLIST STORY xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink">
<!ELEMENT THREADS (EPISODE+)>
<!ELEMENT EPISODE (subgoal, ATTEMPT+, result) >
<!ELEMENT ATTEMPT (action | EPISODE) >
<!ELEMENT INFOS ( ( date | author | a )* )
<!ELEMENT title (#PCDATA) >
<!ELEMENT context (#PCDATA) >
<!ELEMENT problem (#PCDATA) >
<!ELEMENT goal (#PCDATA) >
<!ELEMENT subgoal (#PCDATA) >
<!ELEMENT result (#PCDATA) >
<!ELEMENT moral (#PCDATA) >
<!ELEMENT action (#PCDATA) >
<!ELEMENT date (#PCDATA) >
<!ELEMENT author (#PCDATA) >
<!ELEMENT a (#PCDATA)>
<!ATTLIST a xlink:href CDATA #REQUIRED xlink:type CDATA #FIXED "simple" >
```

4. Defining attributes

Rough syntax of Attribute rules:

```
<!ATTLIST element_name attr_name Attribute_type Type_Def Default >
```

Overview:

Type	Attribute types
CDATA	"Character Data" - Text data
NMTOKEN	A single word (no spaces or punctuations)
ID	Unique identifier of the element.
IDREF	Reference to an identifier.
IDREFS	Reference to one or more identifiers
(A|B|C|..)	List of values (from which the user must choose)

	Type Definition
#IMPLIED	Attribute is optional
#REQUIRED	Attribute is mandatory)
#FIXED Value	Attribute has a fixed value (user can't change it)

- See next slides for examples ....

Examples of attribute definitions:

DTD rule	example XML
<!ATTLIST person first_name CDATA #REQUIRED>	<person first_name="Joe">
<!ATTLIST person gender (male|female) #IMPLIED>	<person gender="male">
<!ATTLIST form method CDATA #FIXED "POST">	<form method="POST">
<!ATTLIST list type (bullets|ordered) "ordered">	<list type="bullets">
<!ATTLIST sibling type (brother|sister) #REQUIRED>	<sibling type="brother">
<!ATTLIST person id ID #REQUIRED>	<person id="N1004">

Shortcut to define multiple attributes for an element:

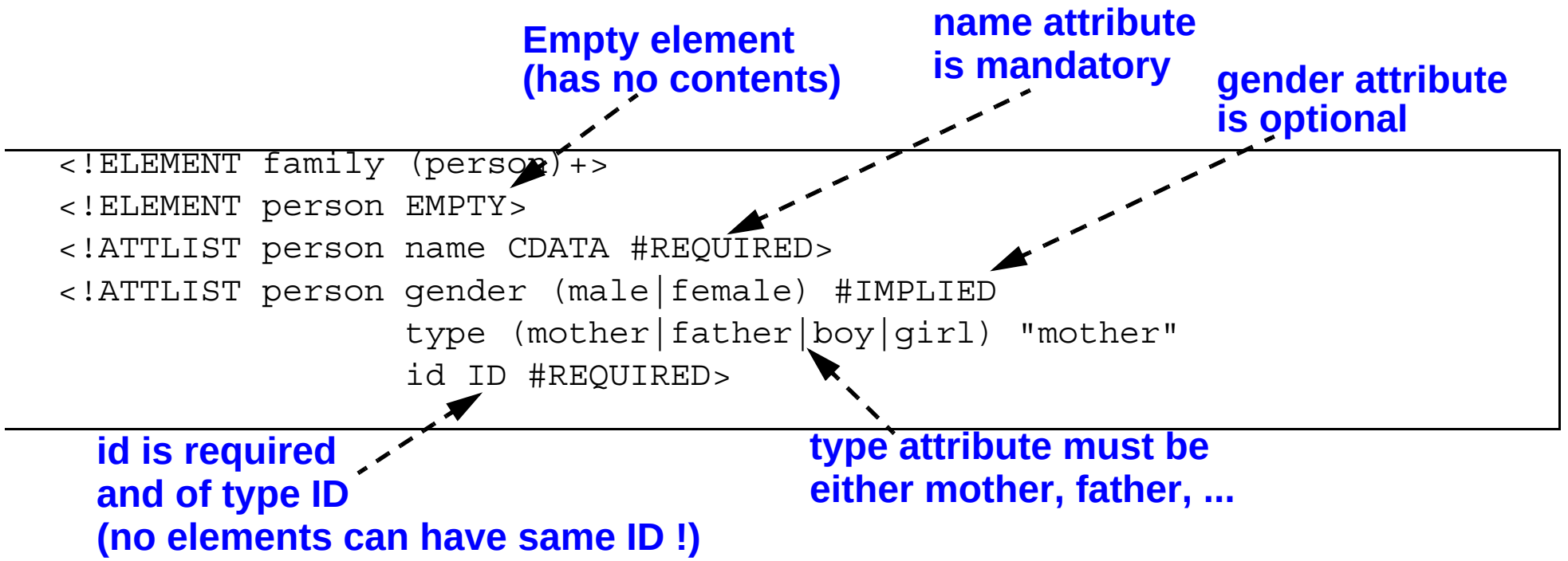
```
Syntax: <!ATTLIST   target_tag
           attr1_nom   TypeAttribut   TypeDef   Defaut
           attr2_nom   TypeAttribut   TypeDef   Defaut
           ...>
```

Shortcut illustrations:

```
<!ATTLIST person
    ident       ID                 #REQUIRED
    gender      male|female)       #IMPLIED
    nom         CDATA              #REQUIRED
    prenom      CDATA              #REQUIRED
    relation    brother|sister)    #REQUIRED >
<!ATTLIST portable
     owner   IDREF              #REQUIRED >
```

Example 4-1: Family DTD

url: http://tecfa.unige.ch/guides/xml/examples/xsd-examples/family.dtd



A valid XML file

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE family SYSTEM "family.dtd">
<family>
  <person name="Joe Miller" gender="male"
          type="father" id="N123456789"/>
  <person name="Josette Miller" gender="female"
          type="girl" id="N123456987"/>
</family>
```

Example 4-2: An address book with ID and IDREF

DTD

url: **http://tecfa.unige.ch/guides/xml/examples/dtd-examples/addressbook_id.dtd**

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT addressBook (person)+>
<!ELEMENT person (name,email*,link)>
<!ATTLIST person id ID #REQUIRED
                 gender (male|female) #IMPLIED>
<!ELEMENT name (#PCDATA|family|given)*>
<!ELEMENT family (#PCDATA)>
<!ELEMENT given (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT link EMPTY>
<!ATTLIST link manager IDREF #IMPLIED
               subordinates IDREFS #IMPLIED>
```

Example XML file:

```
<!DOCTYPE addressBook SYSTEM "addressbook_id.dtd">
<addressBook>
  <person id="B.WALLACE" gender="male">
    <name> <family>Wallace</family> <given>Bob</given> </name>
    <email>bwallace@megacorp.com</email>
    <link manager="C.TUTTLE"/>
  </person>
  <person id="C.TUTTLE" gender="female">
    <name> <family>Tuttle</family> <given>Claire</given> </name>
    <email>ctuttle@megacorp.com</email>
    <link subordinates="B.WALLACE"/>
  </person>
</addressBook>
```

5. Designing a DTD

5.1 Some advice

Don’t forget elements and be liberal

- **Each** element needs to be defined, but only **once** !
- Only make elements mandatory if they really are wanted, else use `element?`

Plan the global structure

- Before you start writing out DTDs, use some simple "language" to draft the structure, e.g. use a notation like:

```
name     ==> family + given
family   ==> "text"
```

In most cases, each "object" of your "information domain" becomes an element

- Use child elements to model components
- Use attributes to describe properties of components

Start from the root element and work your way down:

1. Root element
2. Child elements of root element
3. Child elements of the other elements, etc.

Remember: Each elements is only defined once !

5.2 Attributes vs. Elements

- There are some design rules that may help you decide whether using an element or an attribute
- In case of doubt, always use elements ...

Rather use child elements inside an element (information block):

- if order is important (attributes can’t be ordered)
- if you plan to use the same kind of information block with different parents
- if a future version of DTD may specify sub-components of an information block
- if the information block represents a "thing" (an object in OO programming)
- if the DTD is text-centric, because an author must see contents she/he edits and attributes are often hidden away in XML editors; only use attributes to qualify properties like style !

Rather use attributes for an element

- if an attribute refers to an other element
  - <pet_of owner_name="lisa" pet_type="cat") would refer to <animal category="cat">
- to declare usage/type/etc. of an element:
  <address usage="prof"> ... </address>
- if you wish to list all possible values a user can enter
- if you want to restrict data type of the attribute value (e.g. require a single word)

6. Entities

6.1 General entities

Consider entities as abbreviations for some other content. An entity must be defined in the DTD and its contents are substituted when encountered in the XML file.

- Recall that XML initially only defines 5 entities and that HTML does many more...
- Use the **`< & > &aquot; &apos;`** entities to refer to **`<, &, >,"`** and **`'`**

Syntax of an internal entity definition: **`<!ENTITY entity_name "content">`**

Syntax of an external entity definition: **`<!ENTITY entity_name SYSTEM URI>`**

Syntax of using an entity: **`&entity_name;`**

Illustrations of entity definitions:

DTD rule	XML example	Result
`<!ENTITY jt "Joe Test">`	`<para> &jt; is here<para>`	`<para> Joe Test is here</para>`
`<!ENTITY space " ">`		
`<!ENTITY copyright "&#xA9;">`	`&copyright; D. Schneider`	
`<!ENTITY explanation SYSTEM "project1a.xml">`	`<citation> &explanation; </citation>`	`<citation>` *contents of project1a.xml ...* `</citation>`

6.2 Parameter entities

- Most professional DTDs use parameter entities.
- These are used to simplify DTD writing

```
Syntax: <!ENTITY % entity_name "content">
        <!ENTITY % entity_name SYSTEM "URI">
```

Example 6-1: DTD entities to define reusable child elements

- More complex DTD often use same structures all over. Instead of typing these several times for each element definition, one can use an ENTITY construction like this:

```
<!ENTITY % Content "(Para | List | Listing)*">
```

Later in the DTD we then can have element definitions like this:

```
<!ELEMENT Intro (Title, %Content; ) >
<!ELEMENT Goal (Title, %Content; ) >
```

The XML parser will then simply translate these **`%Content;`** and we get:

```
<!ELEMENT Intro (Title, (Para | List | Listing)*) >
<!ELEMENT Goal (Title, (Para | List | Listing)* ) >
```

Example 6-2: DTD entities to define reusable attribute definitions

- You may use the same procedure to define "bricks" for attribute definitions
- Entity exemple that defines part of an attribute definition

```
<!ENTITY % stamp '
  id ID #IMPLIED
  creation-day NMTOKEN #IMPLIED
  .......
  mod-by NMTOKEN #IMPLIED
  version NMTOKEN #IMPLIED
  status (draft|final|obsolete) #IMPLIED
  approval (ok|not-ok|so-so) #IMPLIED
  main-author CDATA #IMPLIED
'
>
```

ATTLIST definitions below use %stamp;

```
<!ELEMENT main-goal (title, content, (after-thoughts)?, (teacher-comments)?)>
<!ATTLIST main %stamp; >
<!ELEMENT title (...)>
<!ATTLIST main %stamp; >
```

7. A complex text-centric example

Use case description

- Two DTDs that will help students write research projects and reports
- Both include/use a HTML-like DTD to specify formating of element's contents
- Note: Do not worry if you don't understand all the details, but we believe it's important to demonstrate in this course some more "real life" examples

Example 7-1: Student project and paper DTD including a mini XHTML

We define two DTDs:

1. A DTD to define research projects: ePBLproject11.dtd
2. A DTD to structure research papers: ePBLpaper11.dtd

Each of these DTDs only defines top-level "semantic tags" that define the essential structure of each document type.

Elements then contain further markup that is just "stylistic" and taken from HTML

In these DTDs we include the ibtwsh6_ePBL DTD as an external entity

```
<! ENTITY % foreign-dtd SYSTEM "ibtwsh6_ePBL.dtd" >
%foreign-dtd;
```

- For each tag in which we allow stylistic markup, we then can use constructs like:

```
<!ELEMENT introduction %struct.model;>
<!ELEMENT conclusion %struct.model;>
```

7.1 The Itsy Bitsy Teeny Weeny Simple Hypertext DTD

- ibtwsh.dtd is a popular mini XHTML that is used to "fill in" child elements of a semantically structured DTD. Here we present a slightly modified version.

  url: **http://tecfa.unige.ch/guides/xml/examples/dtd-examples/ePBL11/ibtwsh6_ePBL.dtd**

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!--  WARNING: SLIGHTLY MODIFIED FOR USE AT TECFA ! GET THE ORIGINAL
        from http://www.ccil.org/%7Ecowan/XML/ibtwsh6.dtd

 ibtwsh.dtd
 This is the Itsy Bitsy Teeny Weeny Simple Hypertext DTD.
 Its public identifier is -//XML-DEV List//DTD IBTWSH 6.0//EN
 The contents are dedicated to the public domain by
     the author, John Cowan <cowan@ccil.org>, except that
     John Cowan retains the moral right to be known as the author.
-->

<!-- =========== Common attributes =========== -->

<!-- All elements (except full-document elements) have these attributes -->
<!ENTITY % all"
     id       ID    #IMPLIED
     ref      IDREF #IMPLIED
     class    CDATA #IMPLIED
     title    CDATA #IMPLIED">

<!-- All non-empty elements have these attributes -->
<!ENTITY % i18n "xml:lang CDATA #IMPLIED
          dir (ltr|rtl) 'ltr'">
```

```
<!-- =========== Model bricks =========== -->

<!ENTITY % horiz "#PCDATA | a | abbr | acronym | br | cite | code |
     dfn | em | img | kbd | q | samp | span |
     strong | var">

<!ENTITY % vert "address | blockquote | div | dl | h1 | h2 | h3 |
     ol | p | pre | table | ul">

<!-- ========= Models that you can use in your own DTD ============== -->

<!ENTITY % horiz.model "(%horiz;)*">

<!ENTITY % vert.model "(%horiz; | %vert;)*">

<!ENTITY % struct.model "(%vert;)*">

<!-- =========== Horizontal formatting elements =========== -->

<!-- Abbreviations (normal) -->
<!ELEMENT abbr %horiz.model;>
<!ATTLIST abbr %all;>

<!-- Acronyms (normal) -->
<!ELEMENT acronym %horiz.model;>
<!ATTLIST acronym %all;>

<!-- Citation (italics) -->
<!ELEMENT cite %horiz.model;>
```

```
<!ATTLIST cite %all;>

<!-- Source code (monowidth) -->
<!ELEMENT code %horiz.model;>
<!ATTLIST code %all;>

<!--Terms being defined (normal) -->
<!ELEMENT dfn %horiz.model;>
<!ATTLIST dfn %all;>

<!--Emphasis (italics) -->
<!ELEMENT em %horiz.model;>
<!ATTLIST em %all;>

<!--Keyboard input -->
<!ELEMENT kbd %horiz.model;>
<!ATTLIST kbd %all;>

<!-- Quotation (appropriate quotation marks) -->
<!ELEMENT q %horiz.model;>
<!ATTLIST q %all;
 cite CDATA #IMPLIED>

<!-- Sample output text (monowidth) -->
<!ELEMENT samp %horiz.model;>
<!ATTLIST samp %all;>

<!-- Arbitrary span of text -->
<!ELEMENT span %horiz.model;>
<!ATTLIST span %all;>
```

```
<!-- Strong emphasis (boldface) -->
<!ELEMENT strong %horiz.model;>
<!ATTLIST strong %all;>

<!-- Variable names (italics) -->
<!ELEMENT var %horiz.model;>
<!ATTLIST var %all;>

<!-- IMGs added DKS -->
<!ELEMENT img EMPTY>
<!ATTLIST img
  src     CDATA  #REQUIRED
  alt     CDATA  #IMPLIED
  height  CDATA  #IMPLIED
  width   CDATA  #IMPLIED
  align   CDATA  #IMPLIED
  border  CDATA  #IMPLIED
  hspace  CDATA  #IMPLIED
  vspace  CDATA  #IMPLIED
  %all;>

<!-- Hypertext anchors.
 CONSTRAINT: A elements are not allowed inside other A elements, a fact that XML cannot
express. -->
<!ELEMENT a %horiz.model;>
<!ATTLIST a %all;
     href  CDATA #IMPLIED
     name  CDATA #IMPLIED
     rel   CDATA #IMPLIED
     rev   CDATA #IMPLIED
     target (_BLANK | _TOP | _PARENT) #IMPLIED
```

```
>

<!-- Mandatory line breaks -->
<!ELEMENT br EMPTY>
<!ATTLIST br %all;>

<!-- =========== Headers =========== -->

<!ELEMENT h1 %horiz.model;>
<!ATTLIST h1 %all;>

<!ELEMENT h2 %horiz.model;>
<!ATTLIST h2 %all;>

<!ELEMENT h3 %horiz.model;>
<!ATTLIST h3 %all;>

<!-- =========== Lists =========== -->

<!-- Definition list -->
<!ELEMENT dl (dt|dd)+>
<!ATTLIST dl %all;>

<!-- Defined term -->
<!ELEMENT dt %horiz.model;>
<!ATTLIST dt %all;>

<!-- Definition -->
<!ELEMENT dd %horiz.model;>
<!ATTLIST dd %all;>
```

```
<!-- Ordered list -->
<!ELEMENT ol (li)+>
<!ATTLIST ol %all;>

<!-- Unordered list -->
<!ELEMENT ul (li)+>
<!ATTLIST ul %all;>

<!-- List element -->
<!ELEMENT li %horiz.model;>
<!ATTLIST li %all;>

<!-- =========== Basic table support =========== -->

<!-- Shared attributes -->
<!ENTITY % aligns
     "align (left | center | right | justify) #IMPLIED
     valign (top | middle | bottom | baseline) #IMPLIED">

<!-- Table -->
<!ELEMENT table (caption?, tr+)>
<!ATTLIST table
     %all;
     border      CDATA #IMPLIED
     cellpadding CDATA #IMPLIED
     cellspacing CDATA #IMPLIED
     summary     CDATA #IMPLIED
     width       CDATA #IMPLIED>

<!-- Table caption -->
<!ELEMENT caption %horiz.model;>
```

```
<!ATTLIST caption %all;>

<!-- Table row -->
<!ELEMENT tr (th | td)+>
<!ATTLIST tr %all; %aligns;>
<!-- Table header -->
<!ELEMENT th %vert.model;>
<!ATTLIST th %all; %aligns;
     abbr    CDATA #IMPLIED
     axis    CDATA #IMPLIED
     colspan CDATA "1"
     rowspan CDATA "1"
     scope   CDATA #IMPLIED>

<!-- Table data -->
<!ELEMENT td %vert.model;>
<!ATTLIST td %all; %aligns;
     abbr    CDATA #IMPLIED
     axis    CDATA #IMPLIED
     colspan CDATA "1"
     rowspan CDATA "1"
     scope   CDATA #IMPLIED>

<!-- =========== Other vertical elements =========== -->

<!-- Address block -->
<!ELEMENT address %horiz.model;>
<!ATTLIST address %all;>

<!-- Block quotation -->
<!ELEMENT blockquote %struct.model;>
```

```
<!ATTLIST blockquote %all;
     cite CDATA #IMPLIED>

<!-- General text division -->
<!ELEMENT div %struct.model;>
<!ATTLIST div %all;>

<!-- Paragraph -->
<!ELEMENT p %horiz.model;>
<!ATTLIST p %all;>

<!-- Preformatted text -->
<!ELEMENT pre %horiz.model;>
<!ATTLIST pre %all;>

<!-- =========== END OF ibtwsh.dtd =========== -->
```

7.2 Research plan DTD

- ePBLProject11.dtd can be used to describe open ended (exploratory) research project.
- It makes use of ibtwsh.dtd to let authors format contents

  url: **http://tecfa.unige.ch/guides/xml/examples/dtd-examples/ePBL11/ePBLproject11.dtd**

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<!-- ____________________________________________ ____________________  -->
<!-- ePBL-project DTD for student project management & specification    -->
<!--   Copyright: (2004) Paraskevi.Synteta@tecfa.unige.ch         -->
<!--              http://tecfa.unige.ch/~paraskev/                    -->
<!--                          Daniel K. Schneider             -->
<!--              http://tecfa.unige.ch/tecfa-people/schneider.html     -->
```

```
<!--   Created: 13/11/2002 (based on EVA_pm grammar)              -->
<!--   Updated: 07/05/2004        -->
<!-- VERSIONS                      -->
<!-- v1.1 Adaptations to use with Morphon xml editor and addition of IDs-->
<!-- v1.1b fixed vert.model to struct.model / DKS 2004                  -->
<!-- ____________________________________________________________________  -->

<!-- ____________________________ ENTITY DECLARATIONS ____________________________  -->

<!ENTITY % foreign-dtd SYSTEM "ibtwsh6_ePBL.dtd">
%foreign-dtd;

<!ENTITY % id  "id  ID    #IMPLIED">

<!--  ____________________________ MAIN ELEMENT  ____________________________ -->
<!ELEMENT project  (name, authors, date, updated, goal, state-of-the-art, research-
development-questions, methodology, workpackages ) >

<!ELEMENT name  (#PCDATA )>

<!ELEMENT date (#PCDATA )>

<!ELEMENT authors (#PCDATA )>

<!ELEMENT updated (#PCDATA )>

<!ELEMENT goal (title, description )>

<!ELEMENT state-of-the-art  %struct.model;>
<!ATTLIST state-of-the-art  %id;>
```

```
<!ELEMENT research-development-questions (question )+>
<!ELEMENT question (title, description )>

<!ELEMENT methodology  %struct.model;>
<!ATTLIST methodology  %id;>

<!ELEMENT workpackages (workpackage )+>
<!ELEMENT workpackage (planning, objectives, deliverables )>
<!ATTLIST workpackage  %id;>

<!ELEMENT objectives (objective )+>
<!ELEMENT objective (title, description )>

<!ELEMENT deliverables (deliverable )+>
<!ELEMENT deliverable (url, title, description )>
<!ELEMENT url  (#PCDATA )>

<!ELEMENT planning (from, to, hours-of-work?, progress )>
<!ELEMENT from  (#PCDATA )>
<!ELEMENT to  (#PCDATA )>
<!ELEMENT hours-of-work  (#PCDATA )>
<!ELEMENT progress  (#PCDATA )>
<!-- ____________________________________________________________ -->

<!ELEMENT title  (#PCDATA )>
<!ATTLIST title  %id;>

<!ELEMENT description %struct.model;>
<!-- ____________________________________________________________ -->
```

7.3 Research paper DTD

url: http://tecfa.unige.ch/guides/xml/examples/dtd-examples/ePBL11/ePBLpaper11.dtd

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!-- ____________________________________________ ____________________  -->
<!-- ePBL-paper DTD for student project management & specification      -->
<!--   Copyright: (2004) Paraskevi.Synteta@tecfa.unige.ch         -->
<!--              http://tecfa.unige.ch/~paraskev/                     -->
<!--                           Daniel K. Schneider              -->
<!--              http://tecfa.unige.ch/tecfa-people/schneider.html     -->
<!--   Created: 13/11/2002 (based on EVA_paper grammar)             -->
<!--   Updated: 07/05/2004        -->
<!-- VERSIONS                      -->
<!-- v1.1 Adaptation to use with Morphon xml editor and add IDs, IDREFs -->
<!-- v1.1b fixed vert.model to struct.model / DKS 2004                  -->
<!-- SEE ALSO the book.dtd made by DKS that is simply a superset        -->
<!-- ____________________________________________________________  -->

<!ENTITY % foreign-dtd SYSTEM "ibtwsh6_ePBL.dtd">
<!ENTITY % id  "id  ID    #REQUIRED">

<!-- ____________________________ content ____________________________ -->

%foreign-dtd;

<!ELEMENT paper ( info, abstract, (preface)?, introduction, main, conclusion, references,
(annex)?, (aknowledgements)? ) >

<!ELEMENT info  ( title, authors, date, updated, keywords )>
<!ELEMENT title (#PCDATA )>
```

```
<!ELEMENT authors (author )+>
<!ELEMENT author (firstname, familyname, homepageurl, email )>
<!ELEMENT firstname (#PCDATA )>
<!ELEMENT familyname (#PCDATA )>
<!ELEMENT homepageurl (#PCDATA )>
<!ATTLIST homepageurl %all;>
<!ELEMENT email (#PCDATA )>

<!ELEMENT date (#PCDATA )>
<!ELEMENT updated (#PCDATA )>
<!ELEMENT keywords (keyword )+>
<!ELEMENT keyword (#PCDATA )>

<!ELEMENT abstract (#PCDATA )>

<!ELEMENT preface %struct.model;>

<!ELEMENT introduction %struct.model;>

<!ELEMENT main %struct.model;>

<!ELEMENT conclusion %struct.model;>

<!ELEMENT references (reference )+>
<!ELEMENT reference %struct.model;>
<!ATTLIST reference %all;>

<!ELEMENT annex %struct.model;>

<!ELEMENT aknowledgements %struct.model;>
```