Project no. FP6-028038

Palette

Pedagogically sustained Adaptive LEarning Through the exploitation of Tacit and Explicit knowledge

Integrated Project

Technology-enhanced learning

D.INF.02 – First version of Amaya offering a template mechanism

Due date of deliverable: January 31, 2007
Actual submission date: February 19, 2007

Start date of project: February 01, 2006 Duration: 36 months

Organisation name of lead contractor for this deliverable: INRIA

Project co-funded by the European Commission within the Sixth Framework Programme	
Dissemination Level	
PU	Public

Keywords: Document authoring, structured documents, template
Responsible Partner: INRIA

MODIFICATION CONTROL			
Version	Date	Status	Modifications made by
1	02-02-2007	draft	Émilien Kia, Vincent Quint, Irène Vatton
2	12-02-2007	draft	Géraldine Vidou, Yannick Naudet
3	13-02-2007	draft	Liliane Esnault
4	13-02-2007	Final	Vincent Quint

Deliverable manager

Vincent Quint (INRIA)

List of Contributors

Émilien Kia (INRIA), Vincent Quint (INRIA), Étienne Vandeput (ULG), Irène Vatton (INRIA)

List of Evaluators

Liliane Esnault (AESCRA-EM Lyon), Yannick Naudet (CRP-HT), Géraldine Vidou (CRP-HT)

Summary

This document presents the first version of the templating mechanism implemented in the Amaya authoring tool. It briefly introduces the notion of a template and the XTiger language used to represent templates in Amaya. It summarizes the main features for template-driven editing that are introduced in this initial version of the software, as well as the current limitations. Some examples are given. Appendices contain the specification of the XTiger language and the section of the user's manual dedicated to templates.

TABLE OF CONTENT

1. Introduction

Most popular XML document formats used on the web, such as XHTML or SVG, are very flexible: they allow many different types of documents to be represented. This is an advantage in a wide space such as the Web, as a broad range of documents can be handled consistently. XHTML, for instance, is used to represent not only traditional Web pages, but also complex technical documents, sophisticated e-commerce forms or rich media slides, and all these documents can be accessed with a single browser. But this flexibility makes document authoring a complex task. When producing a specific type of document, an author is faced with all the possibilities provided by XHTML, and she has to make a number of difficult decisions. If multiple similar documents have to be produced consistently, for a particular use or for some specific application, authors have to make a consistent use of the XHTML document format, which has proven to be very difficult.

The XTiger language (eXtensible Templates for Interactive Guided Editing of Resources) was created to tackle this problem. It allows a document designer to specify how the document format (XHTML, for instance) has to be used for representing a certain type of document. XTiger relies on the notion of template. A template is a skeleton representing a given type of document, expressed in the format of the final documents to be produced (say XHTML). The format of the final documents is called the *target language* and must be an XML language. The skeleton contains some statements, expressed in the XTiger language, that specify how this minimal document can evolve and grow, while keeping in line with the intended type of the final documents. Some parts of the template may be frozen, if they have to appear as is in the final document. Some parts may be modified when producing the final document, some others may be added either freely or under some constraints. It is the role of the XTiger language to specify these possibilities and constraints.

When a XTiger template is available for a type of document, the role of an authoring tool is to follow the XTiger statements and to help the author to build a document that will be consistent with the rules of its type. The present document introduces a new version of the Amaya authoring tool that implements these principles. It describes the main features of template-driven editing in Amaya, it mentions the limitation of the initial version of the software, and provides a few examples of its use. Appendices provide details about the XTiger language, and the user interface.

The design principles of the XTiger language and the main features of template-driven editing in Amaya are presented in D.INF.01 [1], which defines the architecture of information services in Palette. More information about the XTiger language and its position relatively to other document/web technologies are presented in [2].

The software presented here is part of version 9.54 of the Amaya authoring tool. It can be downloaded from the Amaya web site: `http://www.w3.org/Amaya/User/BinDist.html` Note that the templating feature has to be activated. The procedure is described at: `http://www.w3.org/Amaya/Templates/`

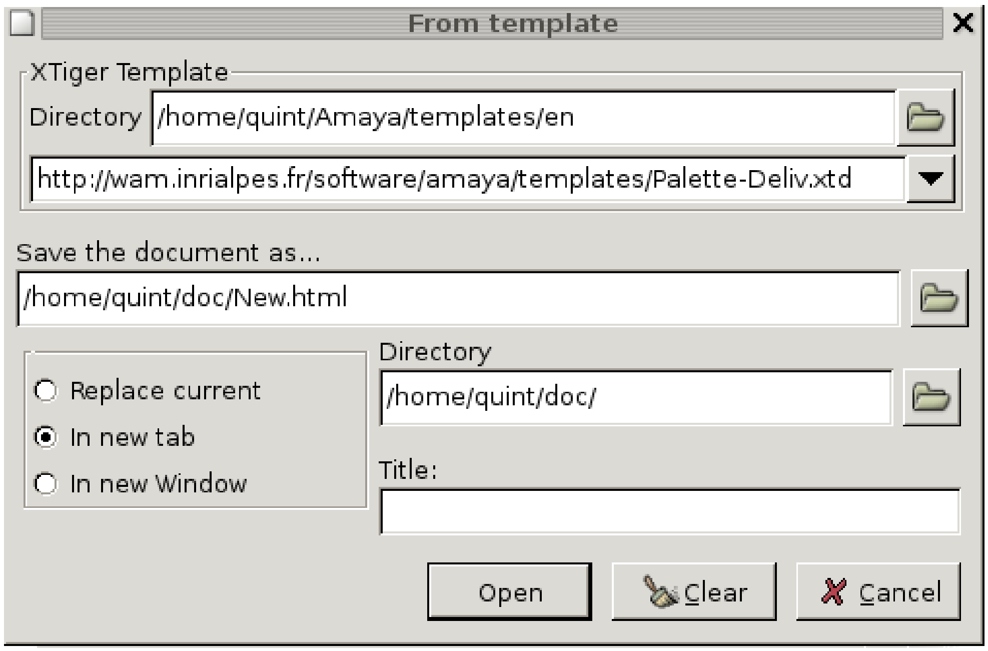
2. Main features

This deliverable is the first release of the Amaya authoring tool that supports template-driven editing based on the XTiger language. Its purpose is to demonstrate the main features of this new editing mode. The implementation is not complete yet, but what is available in this initial release should be enough to make experiments, to collect comments and to prepare the next release, which will offer a broader support for XTiger templates.

In this version, the main features are:

- document instantiation,
- display of XTiger elements,
- repeated structures,
- creation of unions,
- editing in bags.

Document instantiation is the process by which a new document is created from a template. The new document contains a link to its XTiger template and copies of the XTiger elements that will be used by the editor to guide the user. This is achieved by the new command File>New>From_template (Fig. 1). The new document (Fig. 2) can be created on the local file system or on a remote Web server, like any new document in Amaya. Moreover, the template may be a local resource as well as a remote resource available on a web server.


From template

XTiger Template

Directory: /home/quint/Amaya/templates/en

http://wam.inrialpes.fr/software/amaya/templates/Palette-Deliv.xtd

Save the document as...

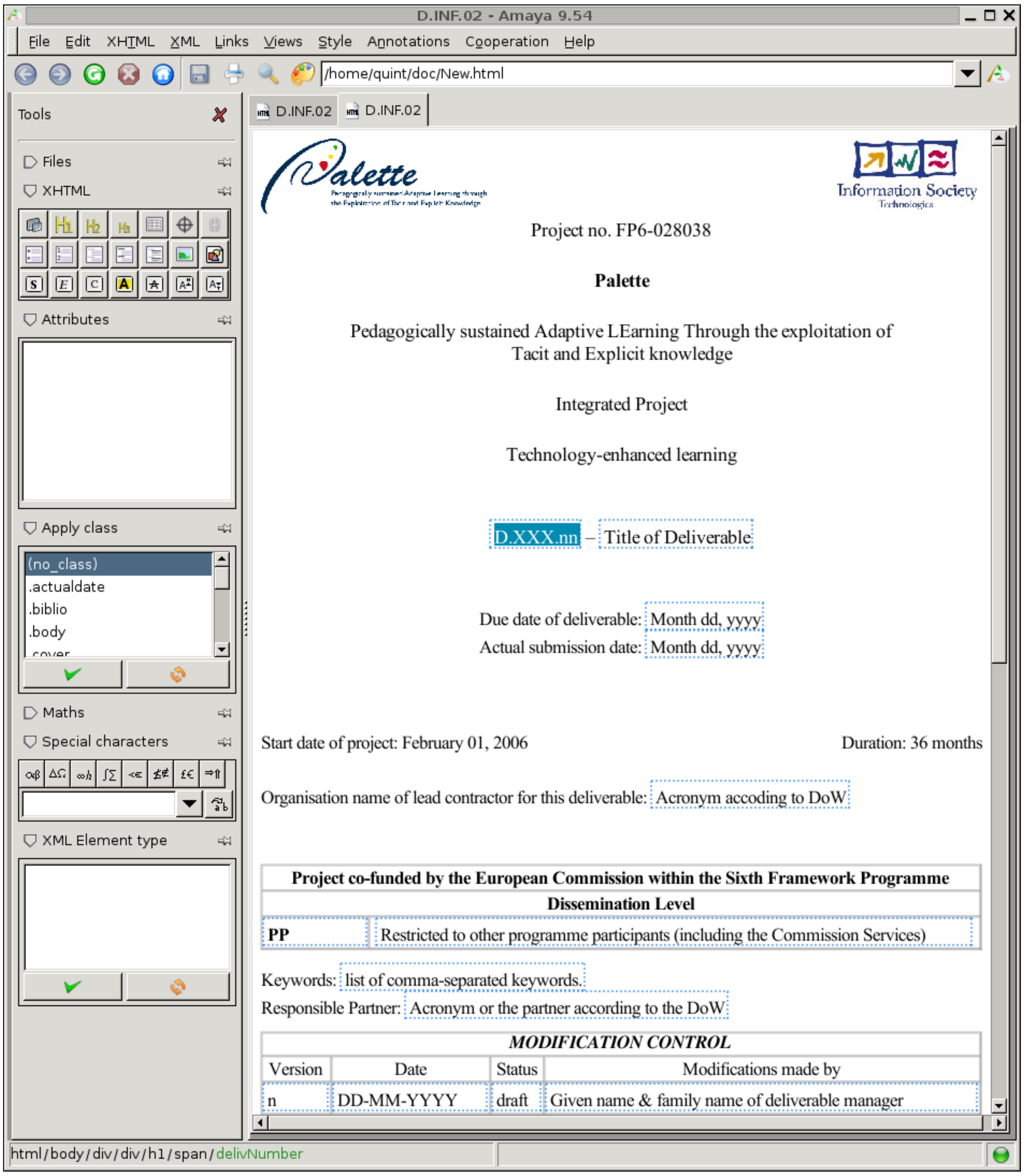
/home/quint/doc/New.html

( ) Replace current
(•) In new tab
( ) In new Window

Directory: /home/quint/doc/

Title:

Fig. 1: Creating a new document from a template



Project no. FP6-028038

Palette

Pedagogically sustained Adaptive LEarning Through the exploitation of Tacit and Explicit knowledge

Integrated Project

Technology-enhanced learning

D.XXX.nn – Title of Deliverable

Due date of deliverable: Month dd, yyyy
Actual submission date: Month dd, yyyy

Start date of project: February 01, 2006 Duration: 36 months

Organisation name of lead contractor for this deliverable: Acronym accoding to DoW

Project co-funded by the European Commission within the Sixth Framework Programme	
Dissemination Level	
PP	Restricted to other programme participants (including the Commission Services)

Keywords: list of comma-separated keywords.
Responsible Partner: Acronym or the partner according to the DoW

MODIFICATION CONTROL			
Version	Date	Status	Modifications made by
n	DD-MM-YYYY	draft	Given name & family name of deliverable manager

html/body/div/div/h1/span/delivNumber

Fig. 2: A new document created from a template

XTiger elements are displayed under the form of colored boxes that delineate the XHTML structure they contain and control (Fig. 3). They are also displayed in the structure view with a different color (green) that allows to clearly separate them from the XHTML structure in which they are embedded. Finally, they have been added, again with the distinctive green color, to the selection path displayed at the bottom of the document window.

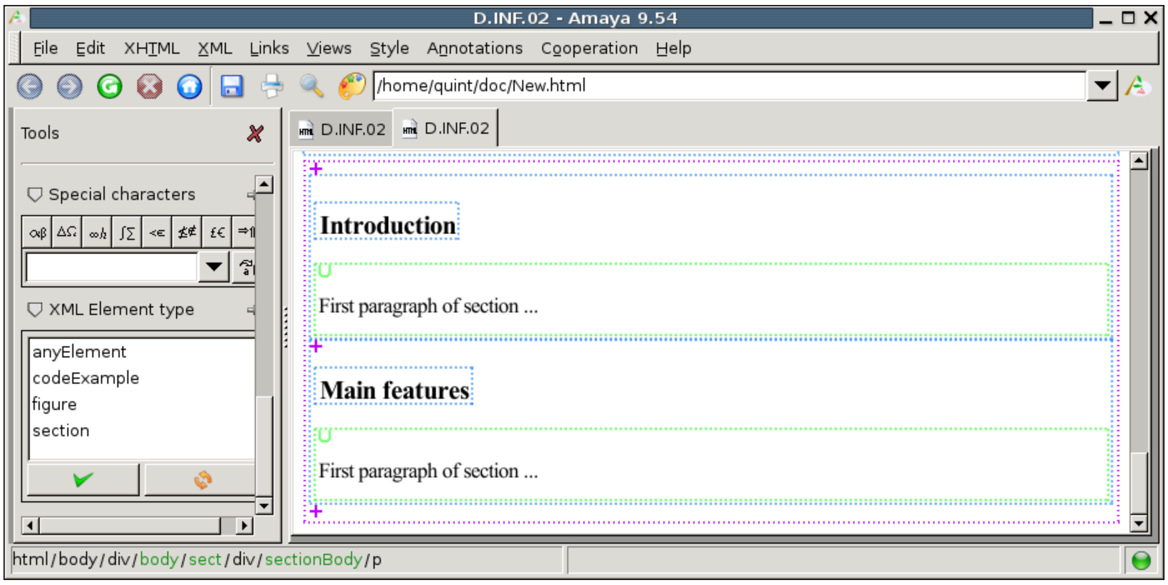

D.INF.02 - Amaya 9.54
File Edit XHTML XML Links Views Style Annotations Cooperation Help
/home/quint/doc/New.html
Tools
Special characters
XML Element type
anyElement
codeExample
figure
section
D.INF.02
D.INF.02
Introduction
First paragraph of section ...
Main features
First paragraph of section ...
html/body/div/body/sect/div/sectionBody/p

Fig. 3: Adding a new component (section) to a repeat element

Repeated structures are the parts of a document that can appear multiple times in sequence at a given position in a document, under the control of a XTiger `repeat` element. Fig. 3 shows two sections (large blue boxes), which are part of a `repeat` element (purple box). This version of Amaya allows an author to easily create and delete repeated structures, while conforming with the constraints (number of occurrences, type of elements) expressed by the `repeat` element. In Fig. 3, clicking the purple '+' icon creates a new section.

Unions introduce some flexibility in a template by offering several possible types for a given element. This is reflected to the user through a pop-up menu presenting all the options defined in the template.

Bags are areas (see green boxes in Fig. 3) that can be edited freely or with a few constraints on the type of usable elements. The constraints expressed in the template are controlled by the editor and the component defined in the template are proposed to the user (Fig. 4).

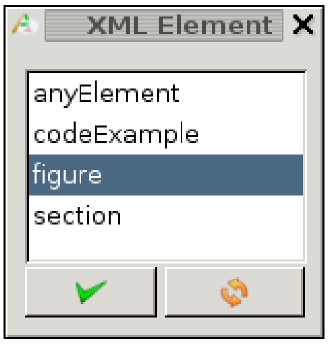


Fig. 4: Creating a new component (a figure) in a section

3. Current limitations

Although the main features for template-driven editing are available, some other, made possible by the XTiger language are still missing, or are not complete:

- attribute element
- optional elements

The `attribute` element of XTiger controls attributes that authors want to associate to XHTML elements. In this initial version, it is simply ignored.

The `option` element from XTiger makes some parts of a document structure optional. Its main functionality is available, but deletion of an existing optional element is not recommended.

XTiger elements are supposed to be used in any part of an XHTML structure. However, in this initial version, it is not recommended to use them in tables or in definition lists (`dl` in XHTML).

Finally, there is no support for creating and editing the XTiger templates themselves. The recommended technique is to use Amaya to create a skeleton XHTML document, and then to use a text editor (or the source view in Amaya) to introduce XTiger elements in the skeleton.

4. References

[1] A. Boukottaya, F. Campoy Flores, R. Deltour, V. Quint, Ch. Vanoirbeek, I. Vatton, K. Zouba, *D.INF.01: Report on the design of extension mechanisms for creating templates, using templates for editing and customizing the user interface, and of extensions to be integrated in the information reuse tool (M3)*, 15 May 2006 (pdf, html)

[2] F. Campoy Flores, V. Quint, I. Vatton, "Templates, Microformats and Structured Editing", DocEng 2006, *ACM Symposium on Document Engineering*, 10-13 October 2006, Amsterdam, The Netherlands, pp. 188-197 (pdf, html)

5. Appendix 1: the XTiger language

When talking about XTiger, it is important to make a distinction between two kinds of documents: a *template* and its *instances*. A *template* is the skeleton presented above, containing XTiger elements and defining a certain type of document. It is the seed used to produce a series of documents, called *instances*, that are derived from the template according to the statements expressed by the embedded XTiger elements. In the rest of the paper, we use the term *instance* instead of final document.

The statements expressed by XTiger elements are supposed to be interpreted by a document authoring tool. Starting from a template, the tool helps the user to follow the XTiger statements, thus ensuring that the instance being edited will stick to the document type specified by the template.

XTiger templates may be used to specify the overall structure of a large document, as well as the fine details of some of its parts. This latter feature allows in particular to express how to use microformats in large documents.

XTiger is not an XML document type like XHTML, SVG or MathML. XTiger is always used in combination with a target language, which is an XML document type. The XTiger elements interspersed in a template are not supposed to be displayed in the same way as elements of the target language. Instead, the role of these XTiger elements is to specify what elements and attributes of the target language must, should or could be present at these positions in the document instance. That is the core of the language, which specifies the structure and (parts of) the content of documents.

This specificity is complemented with additional features that make the language easier to use. For instance, structure fragments can be defined only once and used at several places, in one or several templates. This facilitates a modular construction of templates, by sharing reusable pieces of structure stored in libraries.

As XTiger is used to describe structures, and because it is always mixed with XML languages, it is itself an XML language. XML namespaces are used to distinguish between XTiger elements and elements from the target language. This distinction allows existing web browsers to simply ignore the XTiger elements and to display a template as if these elements were not present.

The XTiger namespace is `http://wam.inrialpes.fr/xtiger`. For the sake of readability, all examples in this document use prefix `xt:` for XTiger element names, while names from the target language are not prefixed.

The target language used in the following examples is XHTML, but it might be any other XML language as well. The first example below is a piece of XTiger language that defines a component called "author". This component is constituted by a XHTML paragraph that contains a few XHTML `span` and `br` elements, with classes from the hCard microformat.

The "author" component can be used to generate the XHTML structure representing an author in document instances, following the hCard microformat.

```
<xt:component name="author">
  <p class="vcard">
    <span class="fn">
      <xt:use types="string" label="name">Author name</xt:use>
    </span>
    <br/>
    <span class="adr">
      <xt:use types="string" label="address">Address ...</xt:use>
    </span>
    <br/>
    <span class="email">
      <xt:use types="string" label="email">email ...</xt:use>
    </span>
  </p>
</xt:component>
```

5.1. Types in XTiger

In XTiger, types are used to specify pieces of structure that may occur at several places in a template or in several templates. XTiger offers a few *basic types* and allows *constructed types* to be built. Constructed types are built with *constructors* that combine XTiger basic types and types from the target language. Two constructors are available: *component* and *union*.

5.1.1. Basic types

XTiger offers three basic types:

- `number` representing integers and floating point numbers,
- `boolean` representing boolean values (`true` or `false`),
- `string` representing variable length character strings.

5.1.2. Target language types

As XTiger always works with a target language and is used to produce documents in that language, it may use elements and attributes from the target language. For instance, when the target language is XHTML, elements `h1`, `h2`, `p`, `strong`, `span`, `cite` are target language types.

5.1.3. Component

`Component` is a constructor that creates a new constructed type by specifying an XML structure assembling other types, which may be basic types, target languages types and constructed types (unions and other components). The type thus created has a name that allows it to be referred from other XTiger elements. This name must be unique in the template where it is defined.

The XTiger element `component` is used to define a component type:

```
<!ELEMENT component ANY>
<!ATTLIST component
    name  NMTOKEN  #REQUIRED>
```

Attributes:

`name`

The name of the type. This attribute must be unique in the template and is mandatory.

Content:

The content of the `component` element defines the structure of the new type. It may be any XML structure that combines target language elements (possibly with attributes) and XTiger elements allowed in the template body.

An example :

```
<xt:component name="hello">
    <p>Hello world!</p>
</xt:component>
```

This example defines a type called "hello" that is a XHTML paragraph (the target language of the template where this elements occurs is XHTML) containing the text *"Hello World !"*. It uses a target language type (`p` element).

5.1.4. Union

`Union` is a constructor that defines a new type as a choice between several types, each of which being a basic type, a target language type, or a constructed type (component or other union). The new type has a name that allows it to be used in other XTiger elements. This name must be unique in the template where it is defined.

The XTiger element `union` is used to define a union type:

```
<!ELEMENT union EMPTY>
<!ATTLIST union
    name    NMTOKEN  #REQUIRED
    include CDATA    #REQUIRED
    exclude CDATA    #IMPLIED>
```

Attributes:

`name`

The name of the type. This attribute must be unique in the template and is mandatory.

`include`

A list of names separated by spaces. These names can be basic types (number, boolean, string), names of elements from the target language (`div`, `h1`, `h2`, `p`, ... for XHTML), or the name attribute of a component or another union. This attribute is used to define the options that constitute the union. This attribute is mandatory.

`exclude`

This attribute is a list of names separated by spaces. These names can be basic types (number, boolean, string), names of elements from the target language (`div`, `h1`, `h2`, `p`, ...), or the name attribute of a component or another union. This attribute is used to exclude some elements that are part of the union as defined by the `include` attribute. This attribute is optional.

Content:

empty.

XTiger provides four predefined unions that may be used in any type definition:

`anySimple`

includes all basic types (`number`, `string` and `boolean`)

`anyElement`

includes all elements defined in the target language.

`anyComponent`

includes all components defined in the template.

`any`

includes `anySimple`, `anyElement` and `anyComponent.`

Example :

```
<xt:union name="hello_or_p" include="hello p"/>
<xt:union name="headings" include="h1 h2 h3 h4 h5 h6"/>
<xt:union name="headings1to4" include="headings" exclude="h5 h6"/>
```

With these definitions, the `hello_or_p` union provides a choice between the `hello` component and the `p` element. The `headings` union provides a choice between all HTML headings (`h1` to `h6`). The `headings1to4` union provides a choice between all HTML headings except `h5` and `h6`.

5.2. Type definitions

The definitions of components and unions presented above must appear in the head of an XTiger template, or in a XTiger library imported by a template.

Type definitions do not appear in document instances. Instead, instances include a Processing Instruction that refers to their template, which contains type definitions and reference to libraries containing additional type definitions.

5.2.1. Head element

The head element collects definitions of components and unions that are used in the template. It also refers to the libraries that contain additional components and/or unions used in the template. This is done with the `import` element.

There is always a `head` element in a template, but only one. It may appear anywhere in the template, but it cannot be the root of the document. In XHTML documents it is recommended to insert it in the XHTML `head` element.

```
<!ELEMENT head ((component | union | import )*) >
<!ATTLIST head
    version         CDATA    #REQUIRED
    templateVersion CDATA    #IMPLIED>
```

Attributes:

`version`

Version of the XTiger language used in the template. For the XTiger language defined in this specification, the value of the version attribute must be "0.8". This attribute is mandatory.

`templateVersion`

Version of the template. A template may evolve over time, when the type of document it represents is modified and several versions are available. This version number may be used to make sure that the right version of the template is used for a given document instance.

Content:

The head element contains any number (including 0) of `component`, `union` and `import` elements, but no other elements.

5.2.2. XTiger libraries

A XTiger library is an XML document containing definitions of constructed types (components and/or unions). Libraries allow types to be declared only once and to be shared between different templates. A XTiger library is defined by the root element `library`. Its content model is the same as the `head` element of a template. Like the `head` element, a library can import other XTiger libraries using the `import` element.

```
<!ELEMENT library ((component | union | import)*)>
<!ATTLIST head
    version         CDATA    #REQUIRED
    templateVersion CDATA    #IMPLIED>
```

5.2.3. Using libraries

When a template or a library uses constructed types defined in a library, that library must be explicitly imported in the template or library that uses it by an `import` element.

```
<!ELEMENT import EMPTY >
<!ATTLIST import
    src    CDATA    #REQUIRED>
```

Attributes:

`src`

URI of the imported library. This attribute is mandatory.

Content:

empty.

All components and unions declared in the imported library are inserted at the position of the `import` element. Some imported components and unions can be redeclared (same `name` attribute) in the current `head` or `library` element. The order of `import` elements in a head or library is important: a `component` or a `union` defined in an imported library with the same `name` attribute as a previous definition replaces that previous definition.

5.3. The template body

A template contains a set of type definitions grouped in the `head` element but also the skeleton of a target language document and some XTiger statements that are used to generate instances. The latter (skeleton and statements) is called the *template body*. A copy of it serves as initial instance when a new document is created from the template. The `head` element and its definitions are not copied in the instance.

All target language elements included in the template body appear in all document instances exactly as they are in the template. Their content is preserved and can not be modified in instances. This is the static part of the template.

There is also a dynamic part in a template, i.e. a part that can be modified under the control of XTiger elements. The XTiger elements that control the dynamic part are:

- `use`, for including elements defined by their type,
- `bag`, for defining free content areas,
- `repeat`, for repeating elements of a given type,
- `option`, for defining an optional part,
- `attribute`, for specifying how to use target language attributes and their values.

5.3.1. Inclusion of types

The `use` element indicates what type(s) of element can appear at that position in an instance. Only one element of the specified type(s) can appear at that position in an instance document.

```
<!ELEMENT use ANY>
<!ATTLIST use
    label       NMTOKEN  #REQUIRED
    types       CDATA    #REQUIRED
    currentType CDATA    #IMPLIED
```

Attributes:

`label`

Label associated with the `use` element. This attribute allows authors of instances to make a difference between the many XTiger elements that appear in a document. It is mandatory.

`types`

A list of space separated names. A single name is also allowed. Each name is either a basic type (`number`, `boolean`, `string`), an element of the target language (`h1`, `h2`, `p`, ...), or the `name` attribute of a component or a union. The element to be inserted at that position in an instance must be of one of these types, but there is no constraint on the descendants of the inserted elements, provided they comply with the DTD or schema of the target language, when target language elements are used. This attribute is mandatory.

`currentType`

Name of the selected type, when a choice has been made. This attribute is present only in instances. It should not be used in a template.

Content:

The `use` element may have a content. If a content is present, it must be of one of the types listed in the `types` attribute. This content is considered as an initial value that will be present in an instance. It may be replaced by an instance author by another content, provided it is compliant with the `types` attribute.

When a component and only that component is used only once in the template, the `use` element may be replaced by a `component` element. This is a shortcut. Its semantics are the same that a `use` element at that position which refers to the component.

Example 1:

```
<xt:use label="birthday" types="string">
Your birth date here
</xt:use>
```

In this example "Your birth date here" is the content that will be displayed when a new instance is created from the template. This string can be freely replaced by an instance author by any other string, but only by a string.

Example 2:

```
<xt:head version="0.8">
  <xt:component name="short_date">
    <xt:use label="day"   types="number">20</xt:use> /
    <xt:use label="month" types="number">10</xt:use> /
    <xt:use label="year"  types="number">1981</xt:use>
  </xt:component>
...
</xt:head>
...
<xt:use label="birthday" types="short_date"/>
```

This example shows how a component can be used to make sure that the user will enter a date in the dd/mm/yyyy format.

```
<xt:use label="date" types="em short_date">
  <em>20 october 1981</em>
</xt:use>
```

Here, the content of the `xt:use` element may be either an XHTML `em` element or a `short_date` component. Only one of them can be inserted at that position in an instance. The current content <em>`20 october 1981`</em> is a valid value, because it is an `em`. It does not need to be *also* a `short_date`.

5.3.2. Free content areas

The `use` element puts strong constraints on the structure and/or content of a part of a document. It is sometimes useful to have more flexibility. That is the role of the `bag` element. It indicates that any number of elements may appear at that position in an instance document, and it specifies the allowed types for these elements.

```
<!ELEMENT bag ANY>
<!ATTLIST bag
    label NMTOKEN  #REQUIRED
    types CDATA    #REQUIRED>
```

Attributes:

`label`

Label associated with the `bag` element. This attribute allows authors of instances to make a difference between the many XTiger elements that appear in a document. It is mandatory.

`types`
: A list of space separated names. A single name is also allowed. Each name is either a basic type (`number`, `boolean`, `string`), an element of the target language (`h1`, `h2`, `p`, ...), or the `name` attribute of a component or a union. The elements to be inserted at that position in an instance must be of one of these types, and all their descendants too. The `types` attribute is mandatory.

Content:
: The `bag` element may have a content. If a content is present, it must follow the constraints set by the `types` attribute. This content is considered as an initial value that will be present in an instance. It may be replaced by an instance author by another content, provided it remains compliant with the `types` attribute.

Example:

```
<p>
  <xt:bag label="para" types="string em strong code">
  This <em>paragraph</em> contains <em><strong>strings</strong></em>
  and <strong><code>any</code></strong> combination of <em>emphasis</em>,
  <code>code</code> and <strong>strong</strong> elements.
  </xt:bag>
</p>
```

5.3.3. Repeated elements

It is often useful to be able to repeat a part of the document structure several times. In this case, the structure to be repeated must first be declared as a component. It can then be used with a `xt:repeat` element in the template body.

```
<!ELEMENT repeat ( use+ | component )>
<!ATTLIST repeat
    label         NMTOKEN #REQUIERED
    minOccurs     CDATA   #IMPLIED "0"
    maxOccurs     CDATA   #IMPLIED "*"
    currentOccurs CDATA   #IMPLIED>
```

Attributes:

`label`
: Label associated with the `repeat` element. This attribute allows authors of instances to make a difference between the many XTiger elements that appear in a document. It is mandatory.

`minOccurs`
: Minimum number of times the component must be repeated. If this attribute is absent, the minimum is 0.

`maxOccurs`

Maximum number of times the component may be repeated. "*" means no upper bound. If this attribute is absent, it is equivalent to "*".

`currentOccurs`

Current number of times the included component is repeated. This attribute is set only in document instances. It should not be present in a template body.

Content:

A `use` element indicates (with its `types` attribute) the type of the component to be repeated. Basic types are not allowed. If the `types` attribute of the `use` element is a list of several types, the repeated elements may have any of these types. Several `use` elements may be present in a `repeat` element in a template to provide initial values to several repeated elements.

Instead of a `use` element, the content may be a single `component` element, when this component is used only there in the whole template. This is equivalent to defining this component in the `head` of the template and putting a `use` element (that refers to that virtual component) in the `repeat` element.

Example:

```
<xt:head version="0.8">
  <xt:component name="bib_item">
    <li>
      <xt:repeat label="authors" minOccurs="1" maxOccurs="5">
        <xt:component name="author">
          <xt:use label="given_name" types="string"/>
          <xt:use label="family_name" types="string"/>
        </xt:component>
      </xt:repeat>
      ...
    </li>
  </xt:component>
  ...
</xt:head>
...
<h2>Bibliography</h2>
<ul>
  <xt:repeat label="bib_list" minOccurs="1">
    <xt:use label="entry" types="bib_item"/>
  </xt:repeat>
</ul>
```

This example describes a bibliography section which includes at least one `bib_item` component. Each of these components contains several authors.

5.3.4. Optional elements

It is often useful to indicate that some part of the document is optional. This is done with the `option` element. This element is equivalent to an `xt:repeat` element with `maxOccurs="1"` and `minOccurs="0"`.

```
<!ELEMENT option ANY>
<!ATTLIST option
    label   NMTOKEN      #REQUIRED
    checked (true|false) #IMPLIED "true">
```

Attributes:

`label`

Label associated with the `option` element. This attribute allows authors of instances to make a difference between the many XTiger elements that appear in a document. It is mandatory.

`checked`

Indicates whether the optional element is present (`true`) or not (`false`). If this attribute is not present, the element is there (`true`). This attribute is usually not present in a template. It is used in document instances.

Content:

The piece of document that is optional.

Example:

```
<xt:option label="bibliography">
  <component name="biblio">
    <h2>Bibliography</h2>
    <ul>
      <xt:repeat label="bib_list" minOccurs="1">
        <xt:use label="entry" types="bib_item"/>
      </xt:repeat>
    </ul>
  </component>
</xt:option>
```

This example defines a component called biblio and makes it optional at the position where it appears in the template body.

5.3.5. Attributes

XTiger provides a way to control attributes from the target language. This is achieved by inserting an `attribute` element as a child of a target language element. The `attribute` element makes an attribute of its parent element mandatory, fixed, or prohibited. If several attributes of a single target language element have to be controlled, several `attribute` elements must be used, one for each of these attributes.

```
<!ELEMENT attribute EMPTY>
<!ATTLIST attribute
    name    NMTOKEN                         #REQUIRED
    type    (number, string, list)          #IMPLIED "string"
    use     (required, optional, prohibited) #IMPLIED "required"
    default CDATA #IMPLIED
    fixed   CDATA #IMPLIED
    values  CDATA #IMPLIED>
```

Attributes:

`name`

Name of the attribute of the parent element that is constrained. This attribute is mandatory.

`type`

Type of the constrained attribute (number, string, list). If the `type` attribute is not present, the default type "string" is assumed.

`use`

Indicates whether the constrained attribute is required, optional, or prohibited. If an attribute is required by its DTD, this attribute will be added even if the template makes it prohibited or optional. If attribute `use` is not present, the default value "required" is assumed.

`default`

Default value of the constrained attribute. This value can be replaced by another value in the instance. Attribute `default` is optional.

`fixed`

Fixed value of the constrained attribute. Attribute `fixed` is optional

`values`

List of possible values. Possible values are separated by spaces. Attribute `values` is optional.

Content:

empty.

Example:

```
<div>
  <xt:attribute name="class" use="optional"
    values="comment example info" default="comment"/>
  ...
</div>
```

This example shows a XHTML `div` element whose `class` attribute is made optional with value limited to the three options `comment`, `example` and `info`. The default value is set to `comment`.

5.4. Resources and processing

When working with XTiger templates, three different kinds of resources are involved:

Template file
A template defines the skeleton of a document and the constructed types (components and unions) it uses. Template files have the `.xtd` extension.

XTiger libraries
Libraries are lists of constructed type definitions. They can be imported by templates and other libraries. Library files have the `.xtl` extension.

Document instances
Instances are documents generated from templates. Instance files have the usual extension of their target language.

When a user creates a document from a template, the new document instance is created as a copy of the template. However, the `xt:head` element with its type definitions is kept by the authoring tool, but it is not copied in the document instance.

The template is linked to the new instance by a processing instruction:

```
<?xtiger        template="URI/of/the/template.xtd"        version="0.8"
templateVersion="xx" ?>
```

which is inserted at the beginning of the instance, in the same way CSS style sheets are linked to XML documents. With this link, the authoring tool can find all the type definitions needed during editing sessions. All other XTiger elements (`use`, `bag`, `repeat`, `option`, `attribute`) as well as all target language elements are kept in the copy that constitutes the initial instance. XTiger types that appear in these elements are replaced by references to their definition in the template (actually, by references to a parsed representation of types in core memory which is more compact).

6. Appendix 2: user manual

This appendix contains the Template section from the Amaya user manual.

6.1. What is a template?

Templates are used to produce multiple documents of the same type. A template is a skeleton representing a given type of document, expressed in the format of the final documents to be produced (XHTML, for instance). The format of the final documents is called the target language and must be an XML language.

In a template, the skeleton document contains some statements, expressed in the XTiger (eXtensible Templates for Interactive Guided Editing of Resources) language, that specify how this minimal document can evolve and grow, while keeping in line with the intended type of the final document. Documents produced from this skeleton following the XTiger statements are called *instances* of the template. Some parts of the template may be frozen, if they have to appear in document instances as they are. Some parts may be changed when producing an instance document, some others may be added either freely or under some constraints.

The file containing a template must have a `.xtd` extension.

A template comes often with a set of accompanying resources (images, style sheets, scripts). A template is a web resource, that can be stored in the local file system or shared on a remote web server.

6.2. Selecting templates

You can create your own templates, using the XTiger language. However, in the current version of Amaya, no specific support is provided for creating or updating templates. You can also use the templates available on the Amaya web site at http://www.w3.org/Amaya/Templates/

You are free to store templates anywhere, in your local file system or remotely.

To ease selection among your favorite templates, there is a Templates section in the Preferences dialogue (Edit>Preferences). Use it to register the templates that Amaya will propose when you create a new document. When registering a local template, you can call a file browser with the browse button to locate and select the files to be added to the list. You can also enter the URI of remote templates in the input area below the list. Use the buttons on the right side of the list to remove a template or to change the order of templates in the list.

Registering templates in this list is not mandatory. When you create a new document instance, you can also choose a template that is not in that list.

6.3. Creating a document instance

To create a new document instance from a template, use entry New>From_template in the File menu. It displays a dialogue box where you can select:

- The directory where the template is located if it is a local file that is not registered in the list of templates
- The template itself, either by selecting in the list of registered templates or by typing the URI or file name of a template that is not in the list
- The URI or file name of the document instance you want to create
- Where to open the new document (Replace current, In new tab, In new window)
- The Title of the new document (this is mandatory)

6.4. Visualizing template elements

When a document instance is displayed, Amaya shows the XTiger elements under the form of colored frames that enclose XHTML elements.

- A `use` element is shown as a dashed blue box
- A `bag` element is shown as a dashed green box
- A `repeat` element is shown as a dashed purple box
- An `option` element is shown as a dashed yellow box
- The `attribute` element is not displayed in the main view, as it only impacts the Attributes menu.

The XTiger elements can also be seen in the structure view. Open this view with the Show_structure entry of the View menu. In the structure view, XTiger elements are displayed in green, while the XHTML elements are in blue. XTiger elements are also shown in the Source view, but with no special color.

Another way to understand the nesting of XTiger elements with XHTML elements is the selection path displayed at the bottom of the document window. There, you can see the list of all elements in the structure of the document from the root element to the current selection. It is updated every time the selection is changed. In this path, XTiger elements are displayed in green and XHTML elements in black. For XTiger elements, it is not the name of the element that is displayed there (use, bag, repeat, option) but the label of these elements, as declared in the template.

6.5. Editing a document instance

Editing is allowed only inside the repeat, option, use, and bag elements, i.e. within the colored dashed boxes. The rest of the document is the fixed part of the template and can not be modified.

6.5.1. Editing in a `repeat` element

In a `repeat` box (purple), you can create a new instance of the repeated structure or you can delete one. This is controlled by the template, which may impose a minimum and/or a maximum number of occurrences of the repeated structure. When creating new occurrences of the repeated structure is allowed, click one of the purple '+' icons to create a new occurrence at that location. You get a pop up menu that tells you what can be inserted at that position. By clicking in that pop-up menu, you create a new occurrence of the corresponding type. In some cases the template offers only one type of occurrence. The pop-up menu then contains a single entry, to clearly state what will be created. If you click outside the menu, nothing is created.

Another way to create new occurrences of the repeated structure is to select an existing occurrence, or to put the caret at the end of an occurrence, and to press the Return key. A new occurrence of the repeated structure is then created, provided the maximum number of occurrences is not reached. The type of the new occurrence is then the same as the one that was selected when pressing the Return key.

When an existing occurrence is empty or fully selected and the Backspace or Delete key is pressed, the element is deleted, provided the minimum number of occurrences is not reached.

6.5.2. Editing in an `option` element

`option` boxes (yellow) display a 'tick' icon in the top left corner. If the box is empty, clicking this icon creates the optional structure. If the optional structure is already present, this removes it.

6.5.3. Editing in a `use` element

Most `use` elements (blue) allow you just to enter free text or to freely replace/edit existing text. Other `use` elements offer a choice between different types of elements that can be inserted at that position. In that case, a triangle icon is displayed in the top left corner of the box. By clicking this icon, you can select one of the allowed types through a pop-up menu. When you have chosen a type, the corresponding structure is generated and you can freely edit its content.

6.5.4. Editing in a `bag` element

In a `bag` box (green), you can insert any number of elements of the types specified by the template. When you want to insert a XHTML element, do it the usual way. It may happen that some elements are not allowed by the template. These elements are greyed out in the menus (in the current version, this feature is not implemented, but prohibited elements are not created).

Templates introduce new structures (called components) built from XHTML elements. The "XML element types" panel lists them and allows you to insert them in bags (green boxes). For inserting a component at the current position, select its name in the panel and click the Apply button at the bottom of the panel. You can also double-click its name.

7. Appendix 3: a sample template

Here is the XTiger template used to edit this document:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!-- This XTiger template defines the structure of a deliverable
     for the FP6 project Palette
-->
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:t="http://wam.inrialpes.fr/xtiger"
      xml:lang="en" lang="en">
<head>
  <meta http-equiv="content-type" content="text/xml; charset=iso-8859-1" />
  <title>Palette FP6-028038</title>
  <link rel="stylesheet" type="text/css" href="Palette-Deliv.css" />

  <t:head version="0.8" templateVersion="1.0">
    <t:component name="figure">
      <div class="figure">
        <img><t:attribute name="id" use="optional"/></img>
        <p><t:use types="string" label="caption">Caption</t:use></p>
      </div>
    </t:component>

    <t:component name="codeExample">
      <div class="example">
        <t:attribute name="id" use="optional"/>
        <t:use types="pre" label="code"><pre>Some code</pre></t:use>
        <p><t:use types="string" label="caption">Caption</t:use></p>
      </div>
    </t:component>

    <t:component name="section">
      <div class="section">
        <h2><t:use types="string" label="heading">Section
            Heading</t:use></h2>
        <t:bag types="anyElement codeExample figure" label="sectionBody">
          <p>First paragraph of section ...</p>
        </t:bag>
      </div>
    </t:component>
  </t:head>
</head>

<body>
<div class="cover">
<div class="logos">
<img alt="Palette logo" src="PaletteLogoWording.png" width="210" style="float: left"
/> <img alt="IST logo" src="ISTlogo.png" style="float: right" /> </div>

<div class="cover-project">
```

```
<p class="projnum">Project no. FP6-028038</p>

<p class="projacronym">Palette</p>

<p class="projname">Pedagogically sustained Adaptive LEarning Through the
exploitation of Tacit and Explicit knowledge</p>

<p class="instrument">Integrated Project</p>

<p class="thematic-priority">Technology-enhanced learning</p>

<h1 class="delivname"><span class="delivNumber"><t:use types="string"
label="delivNumber">D.XXX.nn</t:use></span> – <t:use types="string"
label="title">Title of Deliverable</t:use></h1>

<p class="duedate">Due date of deliverable: <t:use types="string"
label="date">Month dd, yyyy</t:use></p>

<p class="actualdate">Actual submission date: <t:use types="string"
label="date">Month dd, yyyy</t:use></p>

<p class="startdate">Start date of project: February 01, 2006</p>

<p class="duration">Duration: 36 months</p>

<p class="orgname">Organisation name of lead contractor for this deliverable:
<t:use types="string" label="lead">Acronym accoding to DoW</t:use></p>

<table border="1" class="dissemination">
  <tbody>
    <tr>
      <td colspan="2" class="dissemhead">Project co-funded by the European
        Commission within the Sixth Framework Programme</td>
    </tr>
    <tr>
      <td colspan="2" class="dissemlevel">Dissemination Level</td>
    </tr>
    <tr>
      <td class="dissemlev1"><t:use types="string"
      label="code">PP</t:use></td>
      <td class="dissemlev2"><t:use types="string"
          label="dissemination">Restricted to other programme participants
          (including the Commission Services)</t:use></td>
    </tr>
  </tbody>
</table>

<p class="keywords">Keywords: <t:use types="string" label="keywords">list of
  comma-separated keywords.</t:use></p>

<p class="resp-partner">Responsible Partner: <t:use types="string"
label="partner">Acronym or the partner according to the DoW</t:use></p>
```

```
</div>
</div>

<div class="history">
<table border="1">
  <tbody>
    <tr>
      <th colspan="4" class="modif-head">MODIFICATION CONTROL</th>
    </tr>
    <tr align="center">
      <td>Version</td>
      <td>Date</td>
      <td>Status</td>
      <td>Modifications made by</td>
    </tr>
    <tr>
      <td align="center"><t:use types="string" label="version">n</t:use></td>
      <td align="center"><t:use types="string"
      label="date">DD-MM-YYYY</t:use></td>
      <td align="center"><t:use types="string"
      label="status">draft</t:use></td>
      <td><t:use types="string" label="manager">Given name & family name
          of deliverable manager</t:use></td>
    </tr>
    <tr>
      <td align="center"><t:use types="string" label="version">n</t:use></td>
      <td align="center"><t:use types="string"
      label="date">DD-MM-YYYY</t:use></td>
      <td align="center">Final</td>
      <td><t:use types="string" label="manager">Given name & family name
          of deliverable manager</t:use></td>
    </tr>
  </tbody>
</table>
<dl class="people">
  <dt>Deliverable manager</dt>
    <dd><p><t:use types="string" label="manager">Given name & family name
        (Acronym of the partner according to the DoW)</t:use></p>
    </dd>
  <dt>List of Contributors</dt>
    <dd>
      <p>
        <t:use types="string" label="contributors">Given name & family
          name (Acronym of the partner according to the DoW)</t:use>
      </p>
    </dd>
  <dt>List of Evaluators</dt>
    <dd>
      <p>
        <t:use types="string" label="evaluators">Given name & family
          name (Acronym of the partner according to the DoW)</t:use>
      </p>
    </dd>
</dl>

<div class="summary">
<p class="summary-head">Summary</p>
<t:bag types="anyElement" label="summary">
  <p>10 lines maximum</p>
</t:bag>
</div>
</div>

<div class="toc">
<p class="toc-head">TABLE OF CONTENT</p>
<t:bag types="anyElement" label="toc">
   Create table of content here
</t:bag>
</div>

<div class="body">
<t:repeat minOccurs="2" label="body">
  <t:use types="section" label="section"></t:use></t:repeat>
</div>
</body>
</html>
```