Modeling XHTML with UML

Dave Carlson
CTO
Ontogenics Corp.
Boulder, Colorado
dcarlson@ontogenics.com
http://XMLmodeling.com

This document describes the first complete XML Schema for XHTML Basic, which was adopted as a W3C Recommendation in December 2000 [1]. The W3C Recommendation specifies XHTML Basic with a DTD implementation, principally because DTDs were the only recommendation in force at that time. However, we will soon reach a point when the W3C has two schema recommendations, and there are several other XML schema/validation languages that are competing for our attention (RELAX, TREX, and Schematron). Thus, a new approach was taken to produce the XML Schema described here: the XHTML Basic specification was manually reverse-engineered into a Unified Modeling Language (UML) class diagram, then the Schema was automatically generated from that UML model. Other schema languages can be produced in a similar manner; prototypes are under development for generation of DTD and RELAX.

XHTML Basic, as its name suggests, represents the essential core of elements required for presentation of hypertext documents. XHTML Basic was designed to become the document format used by Web clients with limited display capabilities, such as mobile phones, PDAs, pagers, and television settop boxes. In addition to reformulating HTML as valid XML documents, XHTML Basic is also part of a broader effort for the Modularization of XHTML, which decomposes the previous monolithic HTML and XHTML 1.0 specifications into separable, reusable modules [2].

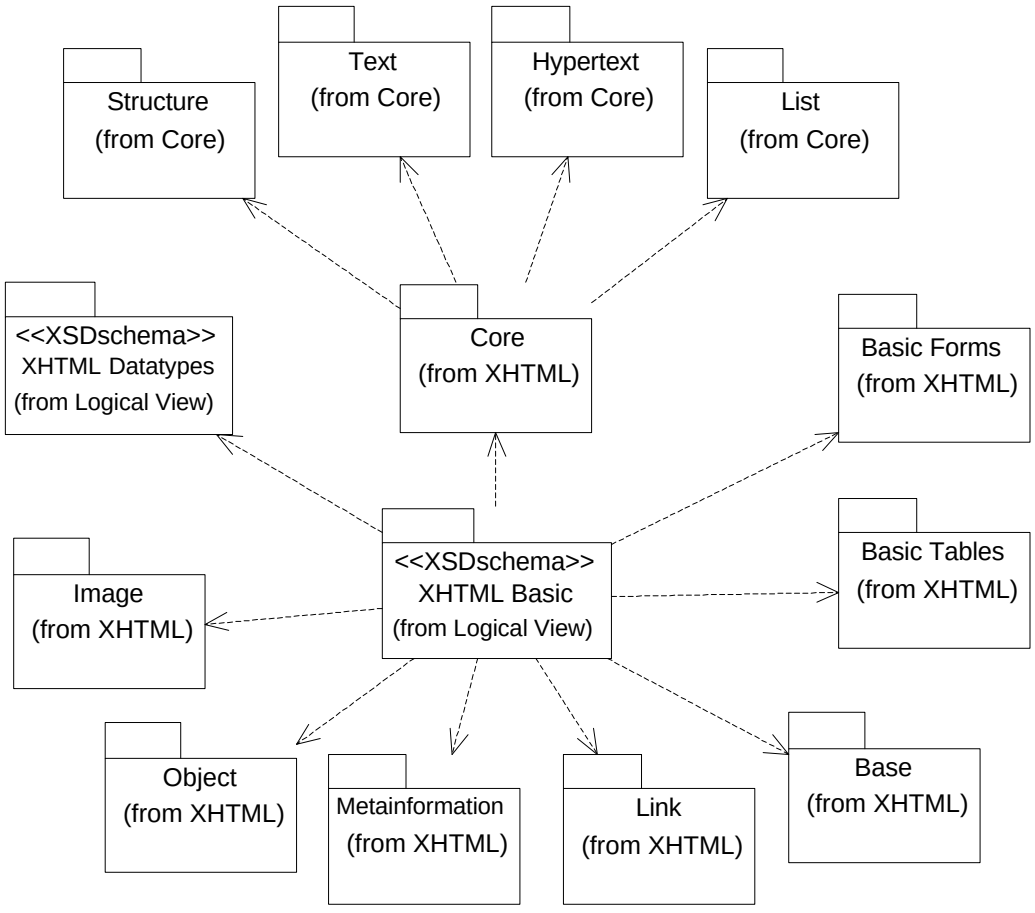
Another useful application involves embedding XHTML content within other XML vocabularies. In fact, it is this requirement that created our original motivation for producing a UML model of XHTML elements. We are using UML to design XML vocabularies such as product catalogs, bibliographies, and e-learning content. In those applications, it's often necessary to support HTML presentation content within other elements; for example, within a product's description or within a mini-tutorial embedded in a training markup language. If XHTML elements such as `<div>`, `<p>`, or `<table>` are available as classes in a UML package, then including them within other vocabularies is a simple matter of drawing an association between classes in a UML diagram. The schema generator takes care of the rest, including generation of the necessary import statements for the XHTML schema definitions.

The focus of the remainder of this document is on presenting the UML model for XHTML Basic. I will not attempt to describe XHTML itself, but instead focus on describing its representation in UML [3]. The XML Schema generated from this model is available as a separate document [4]. For more information on the mapping between UML and XML, refer to my recent book on this subject [5].

XHTML Modularization and UML Packages

The XHTML Modularization specification defines a set of modules that are independent or loosely coupled and that may be combined as necessary to support markup in a particular application. In the example cited previously, where the elements `div`, `p`, and `table` are required, a limited schema can be produced from the Text and Basic Tables modules; all other XHTML markup is not included and therefore invalid for this application. The Text module includes a basic set of elements for headings, blocks, and inline tags.

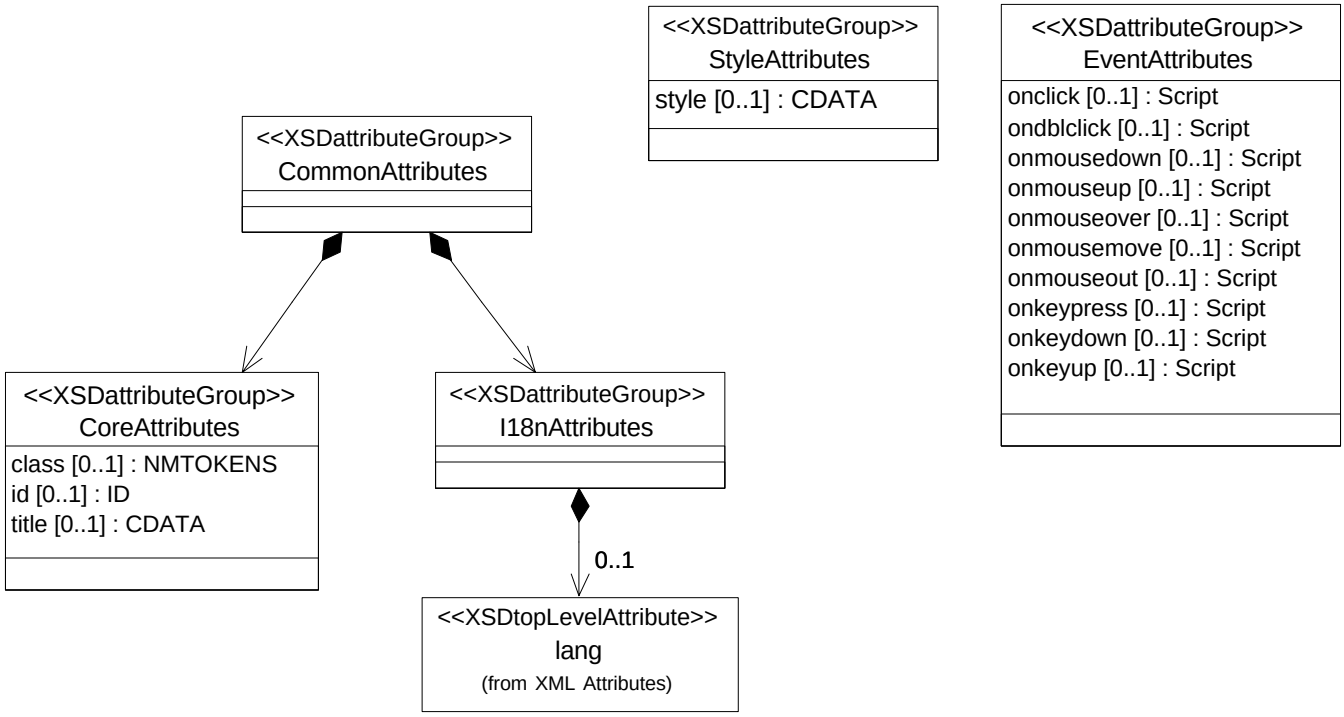
I'm really quite pleased with how well the XHTML modularization mapped into a combination of packages and generalization in the UML model. In the UML, a package defines a namespace for the model elements it contains. The model containing a set of packages may also include dependency relationships between the packages. The full XHMTL Basic package is dependent on eleven packages (modules), plus a set of datatypes defined for XHTML are required by all packages. Four of the packages are further grouped into a Core package. A high-level view of these packages and their dependencies is shown in the following UML package diagram (in a UML diagram, a file folder icon denotes a package).



Attribute Collections

The XHTML Modularization specification defines four attribute groups, which are then selectively aggregated into the CommonAttributes. For XHTML Basic, only CoreAttributes and I18nAttributes are included. These definitions are depicted in the following UML diagram. An XML Schema attributeGroup is defined in UML by adding a stereotype `<<XSDattributeGroup>>` to a UML class. The stereotype mechanism is defined as part of the formal UML specification as a means to extend the UML metamodel for specialized domains. A comprehensive set of UML stereotypes and tagged values are defined in Appendix C of my book [5].

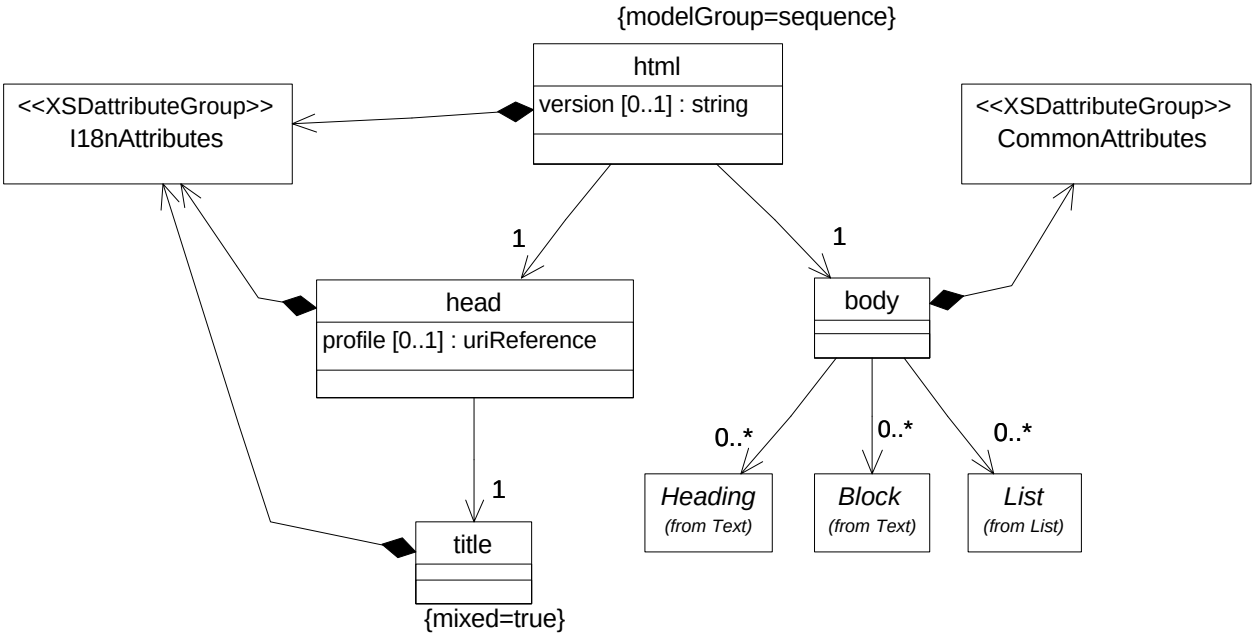
UML models can include multiplicity constraints on either attributes or association ends. An attribute in a UML class is `[1..1]` by default (where `m..n` is interpreted as a pair of min and max values). So in order to override this default, we must specify optional attributes by including the multiplicity `[0..1]` in their definitions. The XML Schema definitions generated from this model are shown following the diagram (for those definitions used by XHTML Basic).




```
<xs:attributeGroup name="CommonAttributes">
   <xs:attributeGroup ref="xhtml:CoreAttributes"/>
   <xs:attributeGroup ref="xhtml:I18nAttributes"/>
</xs:attributeGroup>

<xs:attributeGroup name="CoreAttributes">
   <xs:attribute name="class" type="xsd:NMTOKENS"/>
   <xs:attribute name="id" type="xsd:ID"/>
   <xs:attribute name="title" type="xsd:CDATA"/>
</xs:attributeGroup>

<xs:attributeGroup name="I18nAttributes">
   <xs:attribute ref="xml:lang"/>
</xs:attributeGroup>
```

Structure Module

The XHTML model in UML specifies a default setting that each class will be generated to a schema complexType using a <choice> model group (other models might select <sequence> or <all> as their default). However, the `html` element must use a <sequence> group, so this is specified by adding a tagged value `{modelGroup=sequence}` to the UML class, which is then used by the schema generator. In a similar way, the `title` element must allow mixed content, so a tagged value is used to specify this in the model.



The Schema definitions generated for `html` and `body` are as follows:

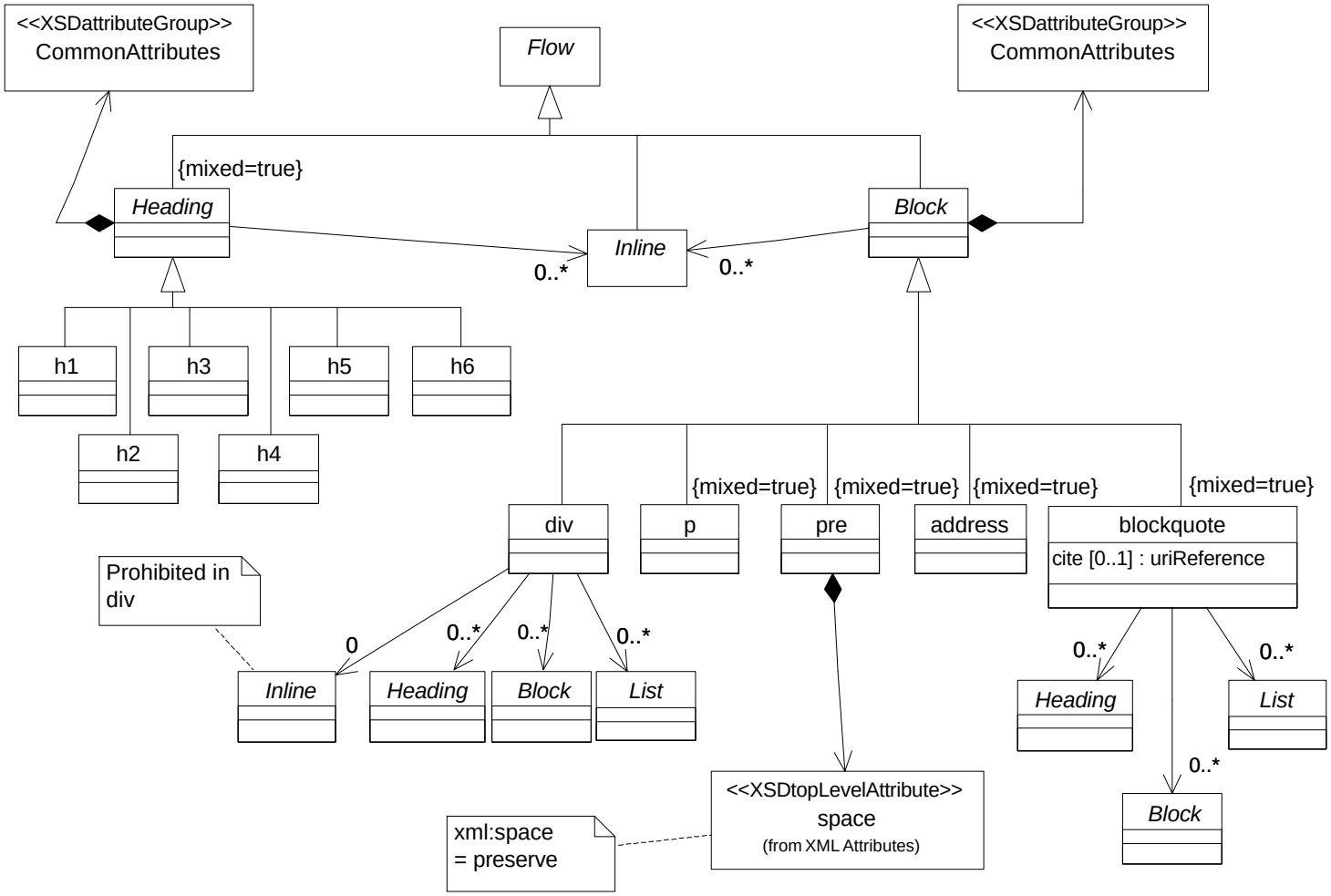
```
<xs:element name="html" type="xhtml:html"/>
<xs:complexType name="html">
   <xs:sequence>
      <xs:element ref="xhtml:head"/>
      <xs:element ref="xhtml:body"/>
   </xs:sequence>
   <xs:attribute name="version" type="xsd:string"/>
   <xs:attributeGroup ref="xhtml:I18nAttributes"/>
</xs:complexType>

<xs:element name="body" type="xhtml:body"/>
<xs:complexType name="body">
   <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="xhtml:Block" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="xhtml:Heading" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="xhtml:List" minOccurs="0" maxOccurs="unbounded"/>
   </xs:choice>
   <xs:attributeGroup ref="xhtml:CommonAttributes"/>
</xs:complexType>
```

Text Module

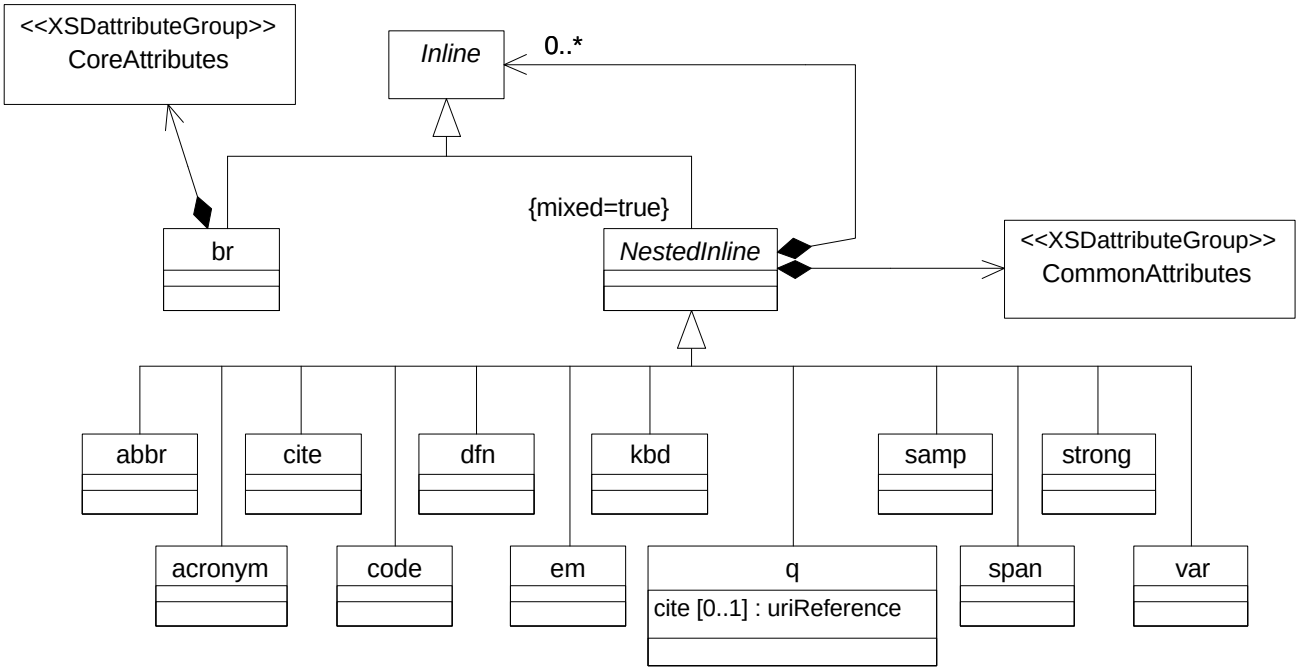
The Text module is by far the largest of all those in XHTML. This module defines four content sets in the W3C specification, named Flow, Heading, Block, and Inline. When mapped to UML, those content sets are modeled as abstract superclasses that generalize the element definitions they contain. The large hollow-headed arrow in UML diagrams represents a generalization relationship, and an abstract class is denoted by a class name in italic font.

This module is defined in two UML class diagrams. The first diagram specifies the first three content sets, and the second diagram specifies the Inline elements. You'll notice one more class name in italics in the first diagram, List, which represents another content set defined in a separate List module.

Note the association from Block to Inline. Because the Inline content set is represented as a superclass generalization in the UML model, then this association allows zero or more instances of any subclass of Inline to be included within a Block. Similar associations are used throughout the remaining module definitions.



The second part of this Text module for Inline elements is represented in the following class diagram. An additional abstract class named NestedInline is added (not part of the XHTML specification) in order to differentiate those elements that may include other Inline elements within their content.



The XML Schema definitions generated from this model could use complexType extension to implement the inheritance specified in the UML model. However, we have had some questionable errors output by validation tools when using extension in this schema, so the following examples are generated without use of extension in the XML Schema. (Our schema generation tool allows extension to be turned on and off with a single configuration parameter. Both types of schemas are available for download on the Web site.)

The Schema definitions for `Flow`, `Block`, and `blockquote` are as follows:

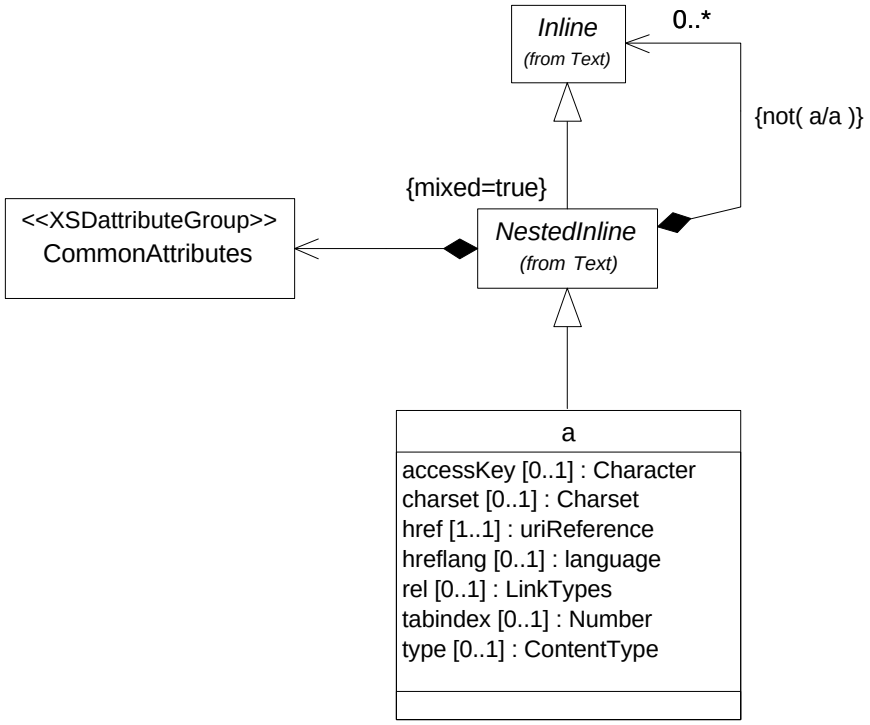
```
<xs:element name="Flow" type="xhtml:Flow"/>
<xs:complexType name="Flow" abstract="true"/>

<xs:element name="Block" type="xhtml:Block" substitutionGroup="xhtml:Flow"/>
<xs:complexType name="Block" abstract="true">
   <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="xhtml:Inline" minOccurs="0" maxOccurs="unbounded"/>
   </xs:choice>
   <xs:attributeGroup ref="xhtml:CommonAttributes"/>
</xs:complexType>

<xs:element name="blockquote" type="xhtml:blockquote"
            substitutionGroup="xhtml:Block"/>
```

```
<xs:complexType name="blockquote" mixed="true">
   <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="xhtml:Inline" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="xhtml:Block" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="xhtml:Heading" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="xhtml:List" minOccurs="0" maxOccurs="unbounded"/>
   </xs:choice>
   <xs:attributeGroup ref="xhtml:CommonAttributes"/>
   <xs:attribute name="cite" type="xsd:uriReference"/>
</xs:complexType>
```

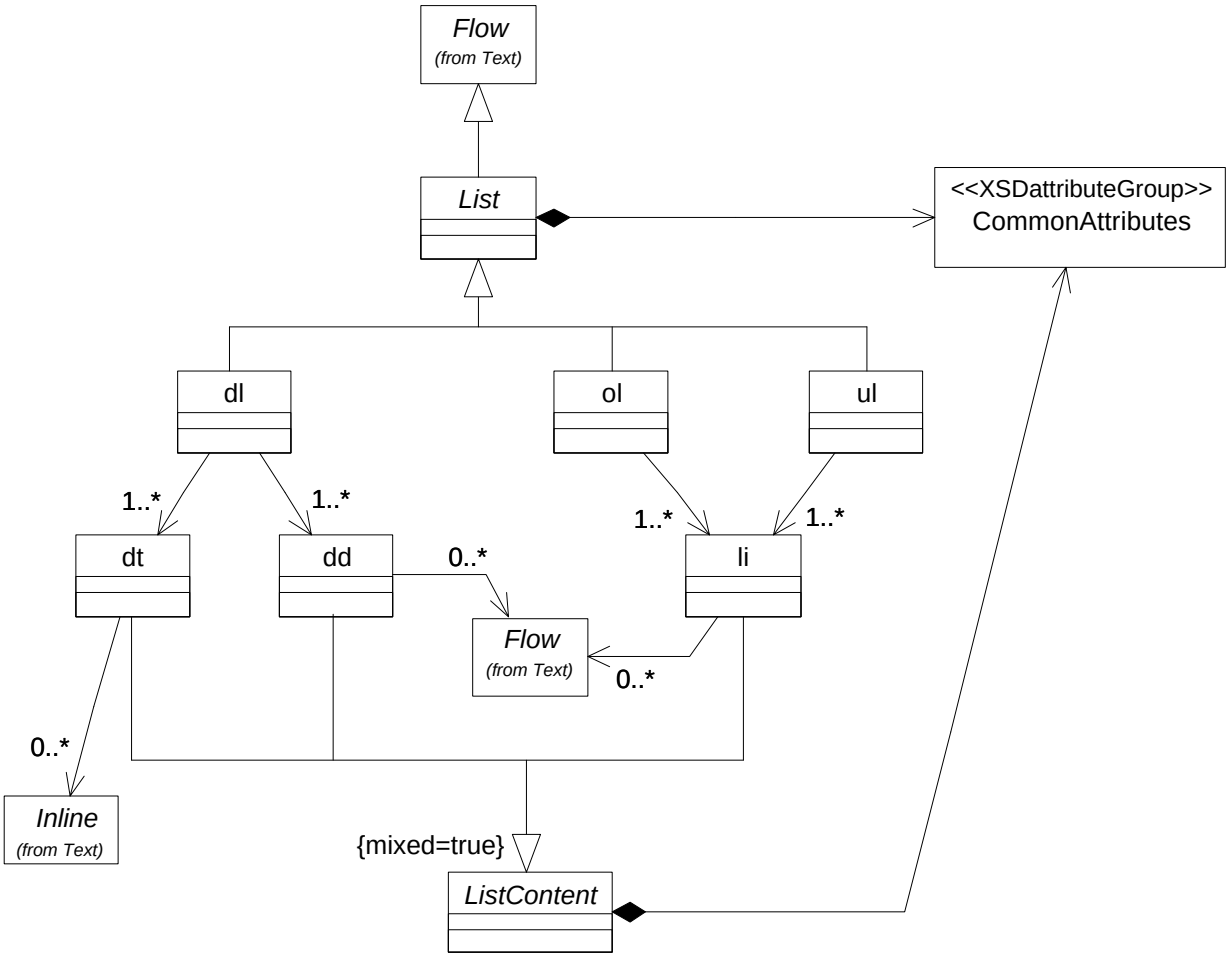
The Schema definitions for `br` and `em` are as follows:

```
<xs:element name="br" type="xhtml:br" substitutionGroup="xhtml:Inline"/>
<xs:complexType name="br">
   <xs:attributeGroup ref="xhtml:CoreAttributes"/>
</xs:complexType>

<xs:element name="em" type="xhtml:em" substitutionGroup="xhtml:NestedInline"/>
<xs:complexType name="em" mixed="true">
   <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="xhtml:Inline" minOccurs="0" maxOccurs="unbounded"/>
   </xs:choice>
   <xs:attributeGroup ref="xhtml:CommonAttributes"/>
</xs:complexType>
```

Hypertext Module

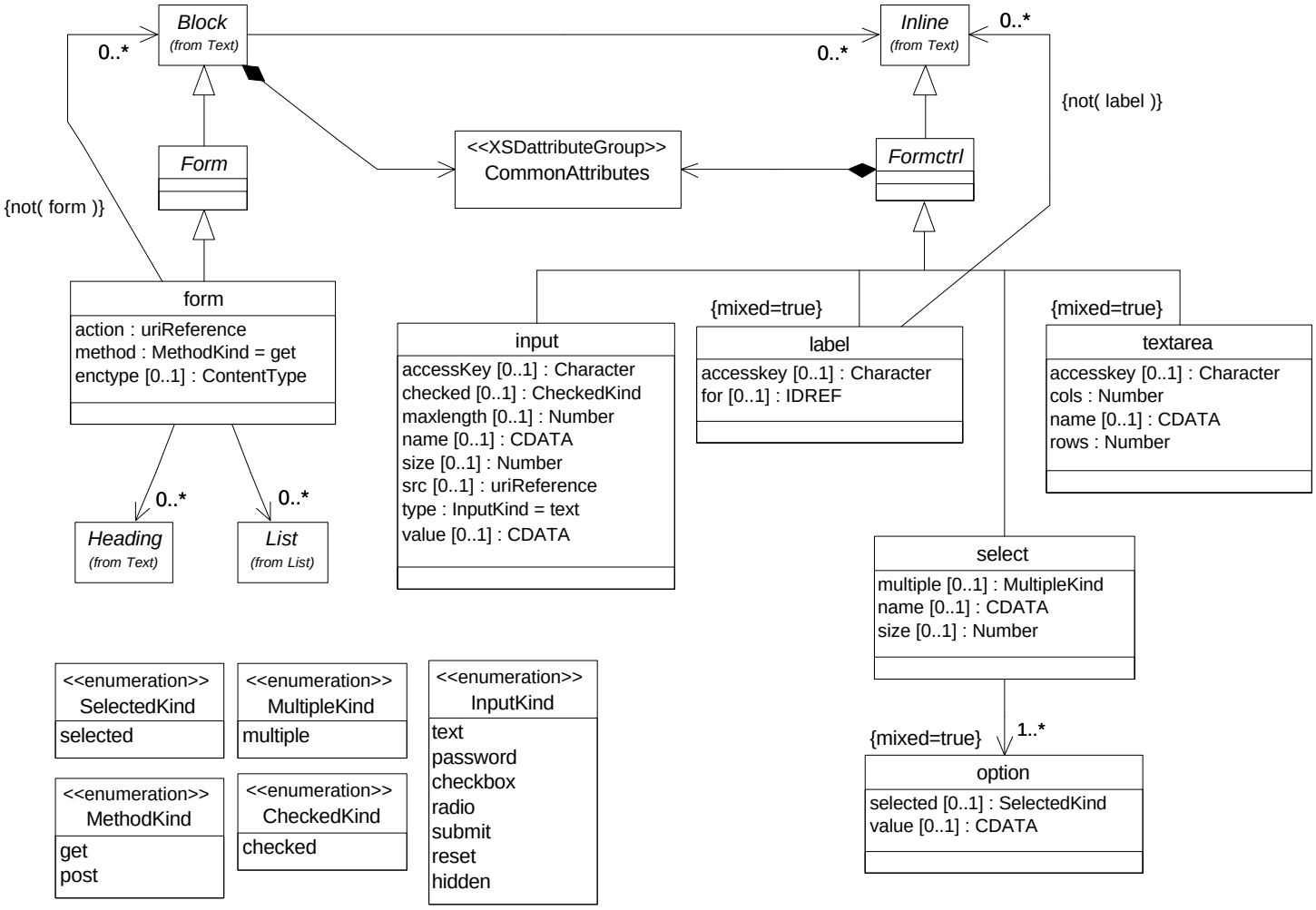


List Module




```
<xs:element name="ul" type="xhtml:ul" substitutionGroup="xhtml:List"/>
<xs:complexType name="ul">
   <xs:choice minOccurs="1" maxOccurs="unbounded">
      <xs:element ref="xhtml:li" minOccurs="1" maxOccurs="unbounded"/>
   </xs:choice>
   <xs:attributeGroup ref="xhtml:CommonAttributes"/>
</xs:complexType>

<xs:element name="li" type="xhtml:li" substitutionGroup="xhtml:ListContent"/>
<xs:complexType name="li" mixed="true">
   <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="xhtml:Flow" minOccurs="0" maxOccurs="unbounded"/>
   </xs:choice>
   <xs:attributeGroup ref="xhtml:CommonAttributes"/>
</xs:complexType>
```

Basic Forms Module




```
<xs:element name="form" type="xhtml:form" substitutionGroup="xhtml:Form"/>
<xs:complexType name="form">
   <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="xhtml:Inline" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="xhtml:Heading" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="xhtml:Block" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="xhtml:List" minOccurs="0" maxOccurs="unbounded"/>
   </xs:choice>
   <xs:attributeGroup ref="xhtml:CommonAttributes"/>
   <xs:attribute name="action" type="xsd:uriReference" use="required"/>
   <xs:attribute name="method" type="xhtml:MethodKind" value="get" use="default"/>
   <xs:attribute name="enctype" type="xhtml:ContentType"/>
</xs:complexType>
<xs:simpleType name="MethodKind">
   <xs:restriction base="xs:string">
      <xs:enumeration value="get"/>
      <xs:enumeration value="post"/>
   </xs:restriction>
</xs:simpleType>
```

Basic Tables Module

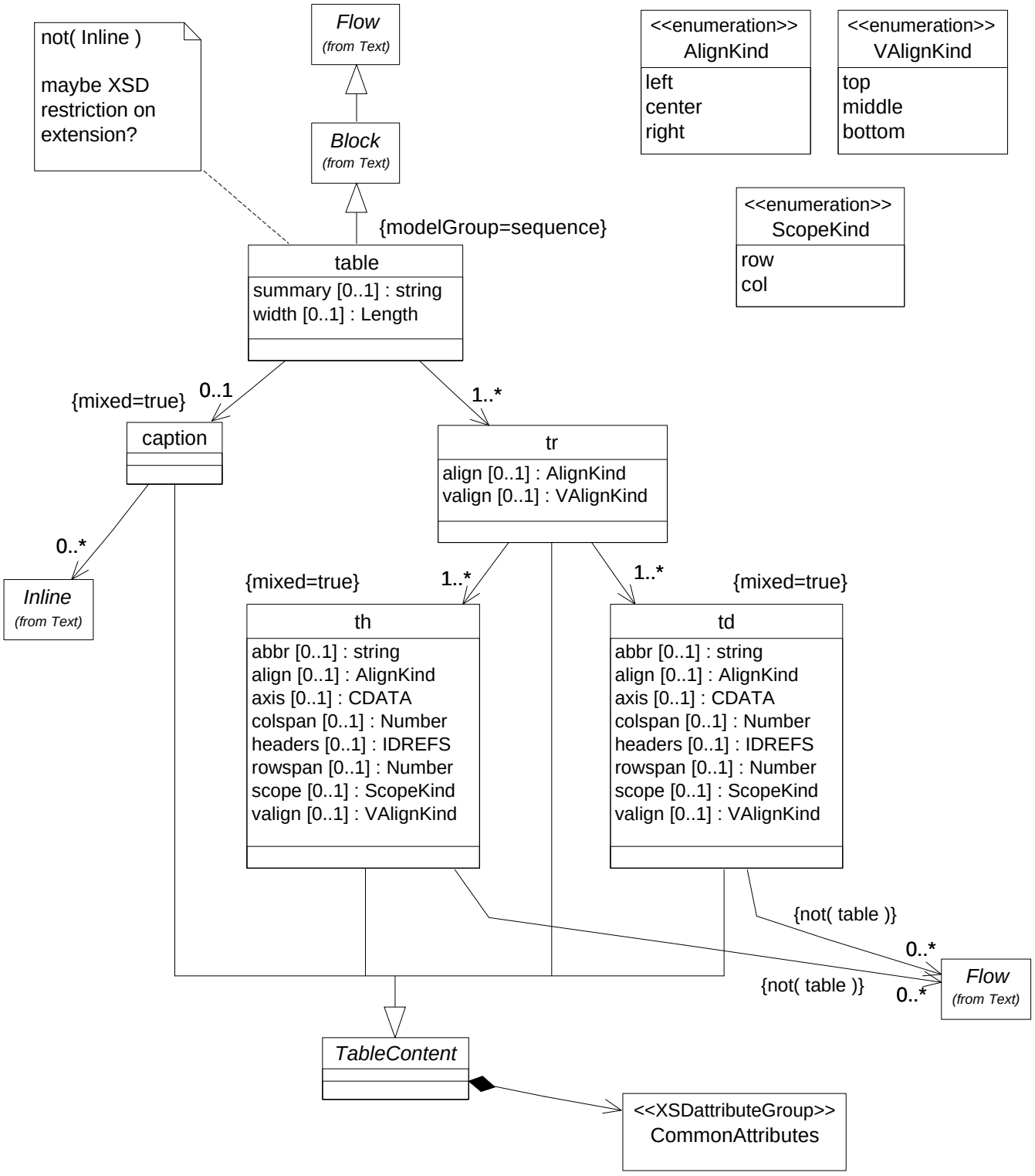
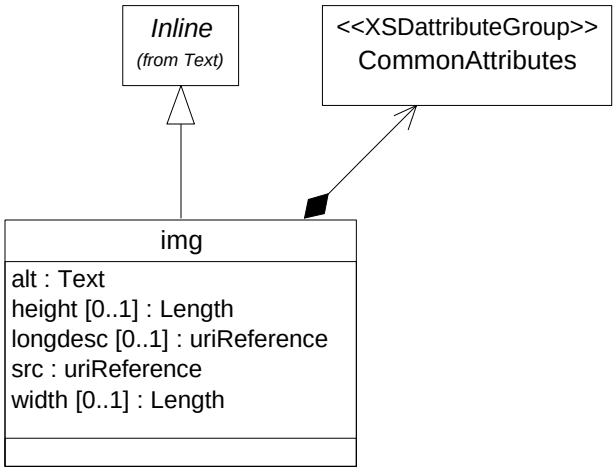
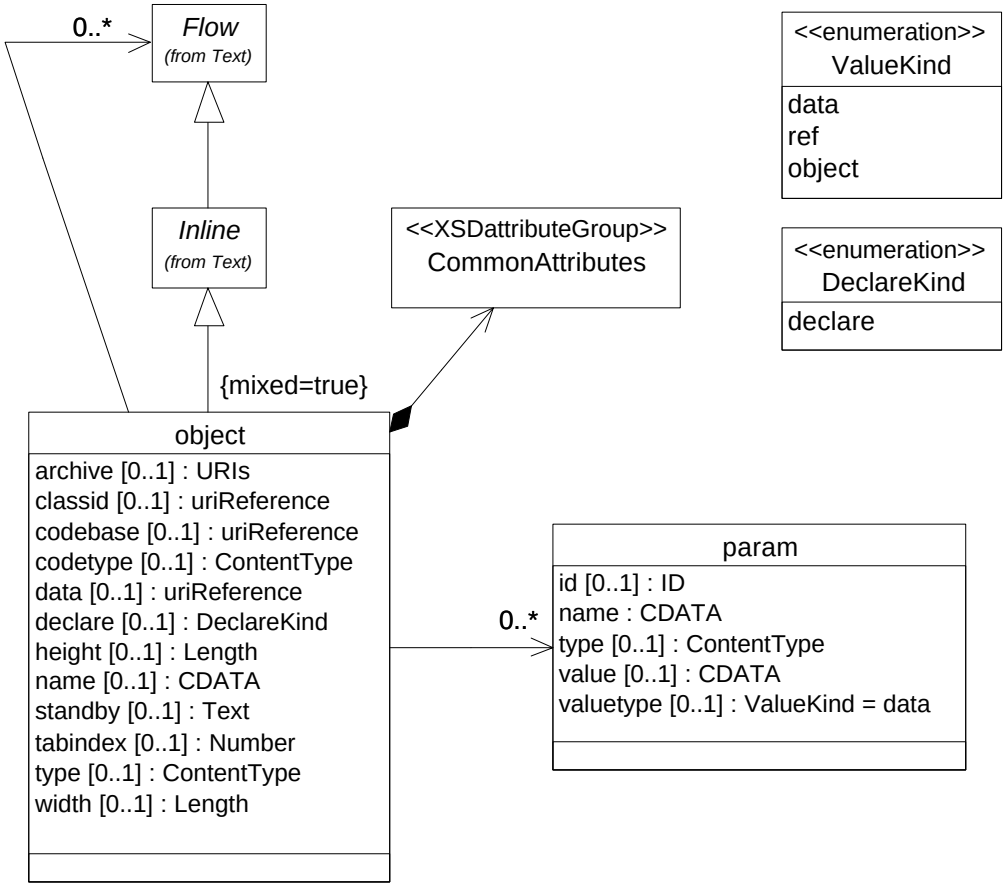


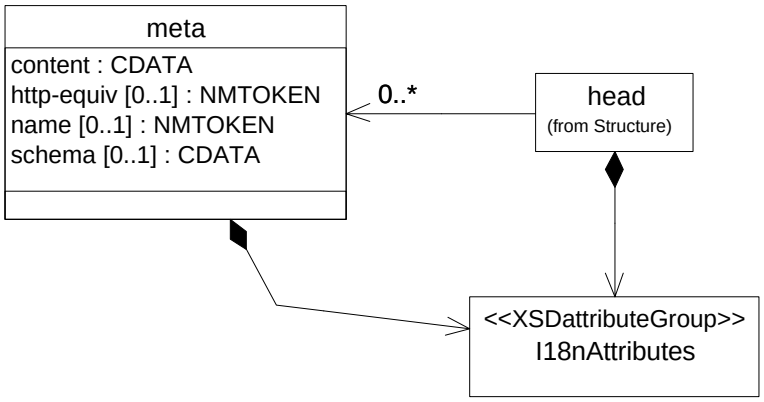
Image Module



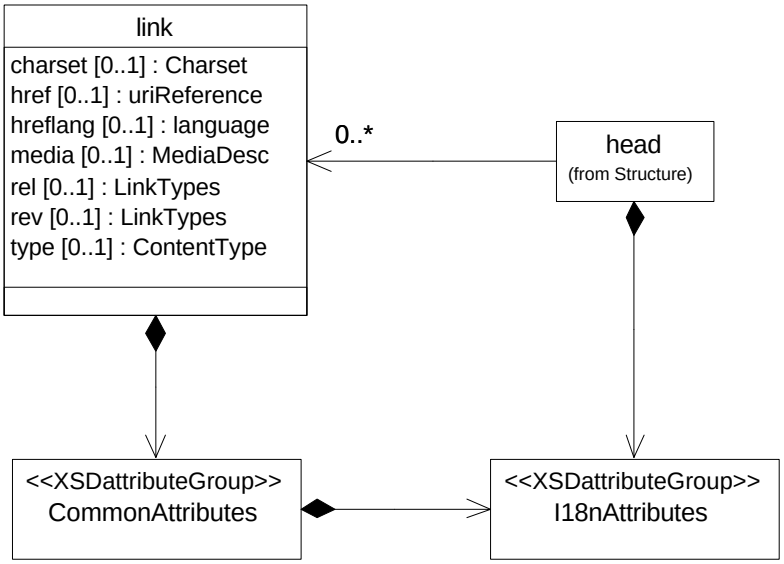
Object Module



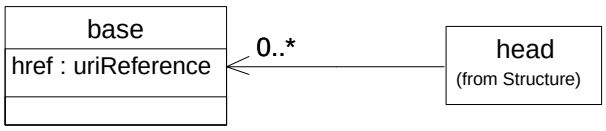
Metainformation Module



Link Module



Base Module



Known Limitations

The schema is currently generated into one large file. A future enhancement to the generator will produce separate schema files for each package (module) in the UML model, controlled by parameter settings.

There are no known omissions in the XML Schema generated from this UML model. There are, however, several places where the schema incorrectly allows child elements.

- The element should not allow Inline elements in its content. The current schema allows this because of inheritance from Block. The UML diagram includes an association from div to Inline with multiplicity [0..0], but this does not restrict the inherited association.
- The same kind of invalid Inline child elements are allowed in the <table> element for the same reason.
- There are several occurrences where an element should not allow nesting of itself (e.g., <a> within <a>). Most of these restrictions are noted on the UML diagrams using constraints on associations, but these constraints are not reflected in the Schema. This issue is similar to the first limitation, where the invalid child elements are inherited. This situation exists for: a, form, label.
- <table> should not be allowed within <th> and <td>, but is allowed because of inheritance from Flow. This nesting would be valid for full XHTML tables without the restriction in XTHML Basic.

Future Enhancements

The following enhancements are required to represent the full XHTML Proposed Recommendation, in addition to the XHTML Basic elements modeled in this version:

- Add remaining module definitions (Text Extension, Frames, etc.).
- Devise a clean approach to insert additional attributes into existing class definitions, as required to support the Intrinsic Events Module, Name Identification Module, and Legacy Module.

References

1. XHTML Basic W3C Recommendation, 19 December 2000. See http://www.w3.org/TR/xhtml-basic
2. Modularization of XHTML W3C Proposed Recommendation, 22 February 2001. See http://www.w3.org/TR/xhtml-modularization
3. For a quick, very accessible introduction to UML and its graphical notation, see: Martin Fowler, *UML Distilled*, 2nd edition, Addison-Wesley, 2000.
4. A Web portal has been created at http://XMLmodeling.com to aggregate newsfeeds and resource references related to modeling XML vocabularies, especially using UML. This site will also contain examples from the book, plus case study examples of modeling XML vocabularies.
5. David Carlson, *Modeling XML Applications with UML: Practical e-Business Applications*, Addison-Wesley, 2001.