Vocabulary ML: Metacode strawman DTD

```
<!-- CC Load File DTD - version 1.0.
     The basic idea for the CC load file is to allow an arbitrary number of 'works' which may be either
Vocabularies or Crosswalks.
     All works have a header and a work ID. Each header has some common material (title, publisher, date,
subject, ...) and some material specific to the type of work it is.
-->

<!-- parameter entities -->
<!ENTITY % idType "ID" >
<!ENTITY % refType "IDREF" >

<!-- commonHeaderBlock
     The CC load file allows us to define a set of works, which are identified chunks of content.  Each work
has a header that gives information on its origin and type. That information is declared in the text macro
below.

    A load file specifies a source vocabulary. Those source vocabularies will almost always be based on an
external source, such as the SIC classification system, or a standard such as Z39.19 (the standard for
thesauri).

    The load file has a header area with fields like Creator, Publisher, Date, Identifier, etc. Those fields are for
describing THE LOAD FILE, not the external source. There is another field, Source, that is used for:
        a) Giving a text citation to the source, such as a standard bibliographic citation.
    or b) Specifying a unique identifier for the source so that a more elaborate description carried elsewhere
can be found.
-->
<!ENTITY % commonHeaderBlock
                  "( dc:Title | dc:Subject | dc:Description | dc:Source
                   | dc:Creator | dc:Contributor | dc:Publisher
                   | dc:Identifier | dc:Language | dc:Coverage
                   | dc:Relation | dc:Date | dc:Type | dc:Format
                   | dc:Rights )*" >

<!-- Title
    Gives the name(s) of the vocabulary.
-->
<!ELEMENT dc:Title (#PCDATA) >
<!ATTLIST dc:Title  xml:lang      CDATA  #IMPLIED>
<!ELEMENT dc:Subject (#PCDATA) >
<!ELEMENT dc:Description (#PCDATA | p)* >
<!ELEMENT p (#PCDATA) > <!-- simple paragraph markup -->
<!ELEMENT dc:Source (#PCDATA) >
<!ELEMENT dc:Creator (#PCDATA) >
<!ATTLIST dc:Creator AgentType CDATA #IMPLIED >
<!ELEMENT dc:Contributor (#PCDATA) >
<!ATTLIST dc:Contributor AgentType CDATA #IMPLIED >
<!ELEMENT dc:Publisher (#PCDATA) >
<!ATTLIST dc:Publisher AgentType CDATA #IMPLIED >
```

```
<!ELEMENT dc:Identifier (#PCDATA) >
<!ATTLIST dc:Identifier IdentifierType CDATA #IMPLIED >
<!ELEMENT dc:Language (#PCDATA) >
<!ELEMENT dc:Coverage (#PCDATA) >
<!ELEMENT dc:Relation (#PCDATA) >
<!ELEMENT dc:Date (#PCDATA) >
<!ELEMENT dc:Type (#PCDATA) >
<!ELEMENT dc:Format (#PCDATA) >
<!ELEMENT dc:Rights (#PCDATA) >
<!ELEMENT CCLoadFile ((SrcVocab | XWalk)*)>
<!ATTLIST CCLoadFile version CDATA #FIXED "0.6">

<!-- SrcVocab
    A SrcVocab is one chunk of content for the concept catalog. It contains a set of "SVTerms", which have
labels, definitions, etc. Those terms come from a "Source Vocabulary".
-->
<!ELEMENT SrcVocab (SVHeader, ForbiddenIDs?, SVTerm+)>

<!-- SVHeader
    A grouping construct so we can easily keep the header material seperate from anything else we might add
in the future.
-->
<!ELEMENT SVHeader (workNum, UIDprefix, displayTitle, briefDisplay,
                %commonHeaderBlock;)>

<!-- UID prefix is the 'friendly' (human-readable) name for a source vocaubulary, such as 'MeSH-1998'. -->
<!ELEMENT UIDprefix (#PCDATA)>

<!-- When reading a source vocabulary, a Concept will be generated for every SVTerm. Those concept IDs
are a 'double-barrelled integer'. workNum is the first integer, which is common across all Concept IDs
generated from the vocabulary, and is not used as the first integer for Concept IDs that come from anywhere
else. -->
<!ELEMENT workNum (#PCDATA)>

<!-- displayTitle is a required field. Its value will be used in situations where the title of the vocabulary will
need to be displayed by the software. -->
<!ELEMENT displayTitle (#PCDATA)>
<!ATTLIST displayTitle  xml:lang      CDATA  #IMPLIED>

<!-- briefDisplay is used in situations where a very brief indication of the vocabulary's identity is needed. -->
<!ELEMENT briefDisplay (#PCDATA)>

<!-- ForbiddenIDs
   Rule #1 is "thou shalt never reassign an ID". It may be the case that something in a work was assigned an
ID, but then deleted. Since we can't reassign IDs, we must track which ones may not be used. That is the role
of the ForbiddenIDs element. If the "serialNumber" attribute is given, that is the last number which is not
safe to reuse. If the serial number is not given, then the element should contain some specification of which
ones can't be reused. Also, the software is assumed smart enough not to reassign an ID that is in the file, so
there is no need to specify the ForbiddenID element if all the IDs that have ever been used are in the file.
-->
<!ELEMENT ForbiddenIDs (any)>
```

```
<!ATTLIST ForbiddenIDs serialNumber CDATA #IMPLIED>

<!-- SVTerm
   This is the basic element in the source vocabulary load module, because this is where we put the
definition of the terms. Every SVTerm must have a UID, which is the unique identifier for the term. UIDs in
general are two strings seperated by a double colon ("::"). However, the first string is given by the UIDprefix
element in the SVHeader, so it is not given in the UID attribute of the SVTerm.

   The UID will usually be an ID that was assigned by the creators of the source vocabulary. If that is not the
case, the UIDtype attribute should be supplied with a value of "other".
-->
<!ELEMENT SVTerm   (CCID?, label, displayLabel*, alt*, syn*,
                    definition*, cla*, parent*, child*,
                    relatedTerm*, typedRelation*,
                    path*, note*, misc*) >
<!ATTLIST SVTerm   UID %idType;  #REQUIRED
                   UIDtype  (source | other) "source" >

<!-- CCID is an optional element that gives the CCID for this term from the source vocabulary. If it is not
given, the system will generate a CCID by using the prefix from the <workNum> element in the header and
sequentially assigning IDs (e.g 107:1, 107:2, ...).  -->
<!ELEMENT CCID (#PCDATA) >

<!-- label
   The 'name' of the term in the classification system. Since labels can, in general, be long, multi-lingual, etc,
this is an element instead of an attribute. The xml:lang attribute is provided in case it is necessary to assign a
language code to the label.
-->
<!ELEMENT label (#PCDATA) >
<!ATTLIST label xml:lang CDATA #IMPLIED>

<!-- displayLabel
    The name that should be displayed. Only needed if the real label is too long, too cryptic, or in a foreign
language. In the content model, display label is repeatable so that it can be given in different languages.
Although the XML DTD can't check it, only one display label should have a particular language code.

    If no language code is given, do we assume "en"?
-->
<!ELEMENT displayLabel (#PCDATA) >
<!ATTLIST displayLabel xml:lang CDATA #IMPLIED>

<!-- alt
    Alternative names and lexical variants of the term such as plurals, abbreviations, etc. The type attributes
(CTYPE, UTYPE) give the CCID (or UID) of the type of variant (e.g. plural), The typeName gives the
human-readable equivalent of the CCID (or UID).
-->
<!ELEMENT alt (#PCDATA) >
<!ATTLIST alt  ctype     CDATA    #IMPLIED
          utype     CDATA    #IMPLIED
          typeName   CDATA    #IMPLIED >
```

<!-- syn
    Text string giving a Synonymous name of the term. This is similar to, but not identical, to the alt element. MeSH has some examples of the use of both alt and syn.
-->
<!ELEMENT syn (#PCDATA) >

<!-- cla
    This is used to give the classification code a source vocabulary may assign to a term independently of its label. Once again, MeSH does this.
-->
<!ELEMENT cla (#PCDATA)>

<!-- definition
    Gives the (natural language) definition of the term. Can be repeated so that the definition can be given in multiple languages.
-->
<!ELEMENT definition (#PCDATA | p)* >
<!ATTLIST definition xml:lang CDATA #IMPLIED>

<!-- The parent and child elements are references to other elements that are hypernyms or hyponyms of the current term.

    If a parent or a child element is supplied, then the presence of the inverse relation is assumed (in other words, if you say A is child of B, you don't have to also say that B is the parent of A, the system will insert those relations for you. However, you are free to specify that if you want).

    Note that there is an interaction with the path element as well - it can specify whole paths in a manner that is redundant with the parent-child relations. At this time we do not have firm recommendations on what is the best way to specify this information.

    The parent and child terms are specified using the UREF attribute. The value of the UREF attribute is the value of the UID attribute of the parent or child SVTerm.

    Note in the content model above that parent can be repeated - one small step towards polyhierarchy in the CC. The parent and child elements have EMPTY content, the only thing they specify is the UREF attribute.
-->
<!ELEMENT parent EMPTY>
<!ATTLIST parent UREF %refType; #REQUIRED>
<!ELEMENT child EMPTY>
<!ATTLIST child UREF %refType; #REQUIRED>

<!-- relatedTerm - Reference to another SV term that is 'related' in the sense of the thesaural relation 'RT'. (In other words, we say that the other term is related to this one but we don't say just how it is related.) -->
<!ELEMENT relatedTerm EMPTY>
<!ATTLIST relatedTerm UREF %refType; #REQUIRED>

<!-- typedRelation
    The <relatedTerm> element handles the vast majority of vocabularies that we have come across so far. There are some vocabularies we know of that not only say that two terms are related, but go on to say just what the relationship is. The <typedRelation> element will let us represent that by holding a CCID that identifies the relationship that holds between two terms (e.g. is-a, inverse-of, antonym, part-of-speech, ...).

Each typedRelation has 2 required attributes. The UREF attribute gives the UID of term that is related to the current term. The other attribute (UTYPE or CTYPE) gives the CCID of the relation that holds between the two terms. (Since a SVTerm is linked to only one Concept at a time, its possible to use the UTYPE attribute to say what Concept to use for the relation by providing the UID for the concept. Alternatively, the CTYPE attribute can be used to give the CCID directly).

The Name attribute can be used to give a human-readable name for the relation, although this could be looked up as the value of the label of the concept identified through CTYPE or UTYPE.

The referenced term can be in this source vocabulary or in an external source vocabulary.
-->
<!ELEMENT typedRelation EMPTY>
<!ATTLIST typedRelation CTYPE CDATA #IMPLIED
UTYPE CDATA #IMPLIED
Name CDATA #IMPLIED
UREF CDATA #REQUIRED>

<!-- path
Specifies the other terms that give paths from the root of the source vocabulary down to this term.

A path is an ordered series of pathComponents, from the root down to the parent of the current term. Each path must have at least one path component. Each pathComponent is an empty element that has a UREF attribute, and an optional human-readable label for the path component. (Normally that label would be obtained from the display label of the corresponding term, but can be supplied here.) PathComponents identify terms from this SV only, they can not refer to terms from other SVs.
-->
<!ELEMENT path (pathComponent)+ >
<!ELEMENT pathComponent EMPTY >
<!ATTLIST pathComponent UREF %refType; #REQUIRED
label CDATA #IMPLIED>

<!-- note
General holder for things like scope notes.
-->
<!ELEMENT note (#PCDATA) >

<!-- misc
General place for putting things that might come in handy in the future.
-->
<!ELEMENT misc (#PCDATA) >

<!-- XWalks (pronounced 'crosswalks') are mappings between source vocabularies. The mappings are carried in a set of Concepts, which hold the terms that are mapped together as having the same meaning. A XWalk has an ID and header like all other kinds of content, followed by a list of Concepts. -->
<!ELEMENT XWalk (XWHeader, Concept*)>
<!ATTLIST XWalk ID CDATA #IMPLIED>

<!-- A XWHeader has a few elements that are specific to it, then a block of elements that are shared with other forms of content.
-->

```
<!ELEMENT XWHeader (workNum, %commonHeaderBlock;)>

<!-- Concept
    The basic element for the concept part of the concept catalog. The ID attribute gives the CCID of the concept. If one is not given then the loading software must generate one, based on the workNum value from the header.
-->
<!ELEMENT Concept   (preferredTerm?, term+, quasiSynonym*, relation*)>
<!ATTLIST Concept   CCID CDATA  #IMPLIED>

<!-- preferredTerm
-->

<!ELEMENT preferredTerm EMPTY >
<!ATTLIST preferredTerm  UREF CDATA #REQUIRED>

<!-- term
    The rest of the source vocabulary terms unified in a single concept go into <term> elements. The value of the UREF attribute needs to be the UID of an SVTerm. METACODE software will have to enforce that constraint, XML can't.
-->
<!ELEMENT term EMPTY >
<!ATTLIST term  UREF CDATA #REQUIRED>

<!-- quasiSynonym
    Some Concepts will be synonymous (i.e. replaceable with) another concept in some but not all situations. Those are called quasi-Synonyms. In this element, the value of the CREF attribute is the ID of another Concept, not a SVTerm. The Concept may come from this XWalk, or from a different one. Since every SVTerm has a unique Concept associated with it, we may also point to that concept indirectly through a UREF attribute.
-->
<!ELEMENT quasiSynonym  EMPTY >
<!ATTLIST quasiSynonym  CREF CDATA #IMPLIED
                UREF CDATA #IMPLIED>

<!-- relation
    Some concepts, such as 'farm', 'farming', and 'farmer' will have a strong relationship, but not one that is synonymy or quasi-synonymy. This element lets us record such relations. The type attributes carry the CCID (or UID or ID) of the type of the relation, which may be the equivalent of 'RT' or may be more specific. The CREF attribute here takes a CCID (not a term ID) as its value.  Unlike the typedRelation element, this is a relation between concepts, not SV terms. As above, we can use CREF or UREF since a term is associated with a unique concept.
-->
<!ELEMENT relation EMPTY>
<!ATTLIST relation CTYPE CDATA #IMPLIED
            UTYPE CDATA #IMPLIED
            Name  CDATA #IMPLIED
            CREF  CDATA #IMPLIED
            UREF  CDATA #IMPLIED>
```