XML Forth Glossary

Here is a rough description of the DocBook Forth Glossary extension I have been considering. This is subject to review.

⟨*glossary*⟩ := <wordset [⟨*id*⟩] [⟨*number*⟩] [⟨*name*⟩] [⟨*core*⟩] [⟨*system*⟩] >
(⟨*worddef*⟩* | ⟨*rationale*⟩* | ⟨*testing*⟩* | ⟨*implement*⟩*)
</wordset>

⟨*id*⟩ := id="⟨*label*⟩"

⟨*number*⟩ := number="⟨*sec-no*⟩"

⟨*name*⟩ := name="⟨*string*⟩"

⟨*core*⟩ := core="⟨*boolean*⟩"

⟨*system*⟩ := system="⟨*label*⟩"

⟨*boolean*⟩ := true | false

⟨*sec-no*⟩ := (⟨*upper*⟩ | ⟨*digit*⟩) (⟨*digit*⟩ | .)*

⟨*label*⟩ := (⟨*upper*⟩ | ⟨*lower*⟩ | ⟨*digit*⟩ | ! | ' | (|) | * | + | , | - | . | / | : | ; | = | @ | [|])+

The <wordset> tag defines a number of defaults as well as encapsulating all of the word definitions (⟨*worddef*⟩), rationale (⟨*rationale*⟩), test cases (⟨*testing*⟩) or implementation (⟨*implement*⟩) for the word set.

id	Label for cross referencing. Note that characters that will cause a problem in one of the intended rendering notations have been forbidden.
number	The location of the word set within the document. The current section number will be by default.
name	The name of the word set (normally a single upper case word)
core	Indicate that the word set describes the core word set (true, the default) or the extended word set (false).
system	The name of the Forth system (or standard) on which this wordset is defined.

```
<wordset id="memory" number="14.6.1" name="MEMORY" system="Forth2012">
```

1 Word definition

⟨*worddef*⟩ := <worddef ⟨*id*⟩ [⟨*number*⟩] ⟨*name*⟩ [⟨*wordset*⟩] [⟨*english*⟩] >
⟨*description*⟩
[⟨*word-rat*⟩]
[⟨*word-test*⟩]
[⟨*word-imp*⟩]
[⟨*see*⟩]
</worddef>

⟨*wordset*⟩ := wordset="⟨*string*⟩"

⟨*english*⟩ := english="⟨*string*⟩"

A glossary definition is enclosed in <worddef> tags. The tag takes four optional arguments:

id	The label used to identify the word in the cross referencing system.
number	The word's number (and sub-number) within the document. The next sequential number will be used by default. This number is appended to the number attribute of the enclosing ⟨*wordset*⟩ (if given).
name	The word name, normally in upper case with special characters escaped.

wordset	The word set this definition appears in. The wordset attribute of the enclosing ⟨*wordset*⟩ will be used by default.
english	The pronunciation for the word.

```
<worddef id="core:DOES" number="1250" name="DOES&gt;" english="does">
```

Question:

In the LaTeX, id is optional, the default label is made by separating the id attribute of the enclosing <wordset> and name attributes with a colon. The id attribute is only provided to override the name attribute when that uses a character that can not be types easily. Thus, if the id attribute of the enclosing <wordset> is "core" we would have:

name="IF" ⇒ *core:IF*
name=">" id="more" ⇒ *core:more*

Should this behaviour be reproduced in the XSLT?.
Or, to put it another way, should the id attribute be optional?

1.1 Description

The main description of the operation of the word. The description can be broken down into different ⟨*section*⟩s.

⟨*description*⟩ := <description>
([⟨*interpret*⟩] [⟨*compile*⟩] [⟨*init*⟩] [⟨*execute*⟩] [⟨*runtime*⟩] ⟨*para-text*⟩*) |
[⟨*sec-text*⟩]
</description>

⟨*interpret*⟩ := <interpret> ⟨*sec-text*⟩ </interpret>

⟨*compile*⟩ := <compile> ⟨*sec-text*⟩ </compile>

⟨*init*⟩ := <init> ⟨*sec text*⟩ </init>

⟨*execute*⟩ := <execute [⟨*type*⟩]> ⟨*sec text*⟩ </execute>

⟨*runtime*⟩ := <runtime [⟨*type*⟩]> ⟨*sec text*⟩ </runtime>

⟨*sec-text*⟩ := ⟨*para-text*⟩*

A <description> should include a general description of the word which may be broken down into further sections.

1.2 Rationale

If the word requires additional discussion, that is not part of its definition, this discussion should be included in the rationale, this should include the reasoning for including the word in the word set. The rationale for a word can be provided in two places. It can be included with the word definition (inside a <worddef>) or the rationale for all of the words in a word set can be collected together (inside a <wordset>).

⟨*word-rat*⟩ := <rationale> ⟨*sec-text*⟩ </rationale>

⟨*att-def*⟩ := ⟨*id*⟩ [⟨*number*⟩] ⟨*name*⟩ [⟨*wordset*⟩]

⟨*rationale*⟩ := <rationale ⟨*att-def*⟩ > ⟨*sec-text*⟩ </rationale>

When defined within a word definition (<worddef>) the ⟨*word-rat*⟩ element is used. This requires no additional information as the word definition provides a context for the rationale.

When defined within a separate collection of rationale's the ⟨*rationale*⟩ element is used. The definition context is not available, thus a number of additional attributes are used to provide this context. The attributes have the same function as for the ⟨*worddef*⟩.

Question:

In the LaTeX, the label for a rationale is constructed by adding the text "rat:" to the start of the word's label. Thus if the word's label is "core:IF" the label for the rationale would be "rat:core:IF". The label is constructed in the same way as for ⟨worddef⟩, thus the id attribute has the same function.

Therefore the same question applies, should this behaviour be replicated in XSLT?

The element ⟨*rref*⟩ is used to cross-reference the rationale section of a word's definition.

When processing the file the first pass extracts the ⟨*word-rat*⟩ elements, collating them in a <wordset> for inclusion in the appendix. Subsequent processing ignores the ⟨*word-rat*⟩ element.

1.3 Testing

The unit testing for a word should be given in the testing section. As with <rationale>, this can be included with the word's definition (⟨*word-test*⟩) or collated into a list of test cases (⟨*testing*⟩).

⟨*word-test*⟩ := <testing> (⟨*sec-text*⟩ | ⟨*test*⟩)* </testing>

⟨*testing*⟩ := <testing ⟨*att-def*⟩ > (⟨*sec-text*⟩ | ⟨*test*⟩)* </testing>

The "test:" prefix is added to the word's label to generate the label for the reference implementation. The ⟨*tref*⟩ element is used to reference an implementation.

In addition to normal text (⟨*sec-text*⟩) the <test> tag can be used to describe an individual test case.

⟨*test*⟩ := <test [type="(X | R)*"]>
<pre>⟨*inline-source*⟩</pre>
<post>⟨*inline-source*⟩</post>
</test>

For example,

```
<test type="XX"><pre>0 DUP</pre><post>0 0</post></test>
```

describes the following test:

```
T{ 0 DUP -> 0 0 XX}T
```

The type argument is used to provide the type of the parameters required for the floating-point test harness.

1.4 Implement

Provides a reference implementation. As with <rationale>, this can be included with the word's definition (⟨*word-imp*⟩) or collated into a list of implementations (⟨*implement*⟩).

⟨*word-imp*⟩ := <implement> ⟨*multi-line-source*⟩ </implement>

⟨*implement*⟩ := <implement ⟨*att-def*⟩ > ⟨*mulit-line-source*⟩ </implement>

The "imp:" prefix is added to the word's label to generate the label for the reference implementation. The ⟨*iref*⟩ element is used to reference an implementation.

1.5 Cross referencing

Cross references to other words should be included in the <see> section. This consists of a collection of ⟨*wref*⟩, ⟨*rref*⟩, ⟨*tref*⟩, ⟨*iref*⟩ and ⟨*xref*⟩ tags.

⟨*see*⟩ := <see> ⟨*cross-ref*⟩* </see>

⟨*cross-ref*⟩ := ⟨*wref*⟩ | ⟨*rref*⟩ | ⟨*tref*⟩ | ⟨*iref*⟩ | ⟨*xref*⟩

⟨*att-ref*⟩ := word="⟨*string*⟩" [⟨*wordset*⟩]

⟨*wref*⟩ := <wref ⟨*att-ref*⟩ />

⟨*rref*⟩ := <rref ⟨*att-ref*⟩ />

⟨*tref*⟩ := <tref ⟨*att-ref*⟩ />

⟨*iref*⟩ := <iref ⟨*att-ref*⟩ />

⟨*xref*⟩ := <xref label="⟨*string*⟩"> ⟨*text*⟩ </xref>

The cross referencing functions <wref>, <rref>, <tref> and <iref> all follow the same pattern. The word attribute is the label of the word being referenced. They look in the current word set for the word, if not found they will look for the word in the core word set. The optional wordset attribute can be used to override the current word set context. When the word is found it will output the reference number and the name, making them both link to the definition being referenced. Thus:

```
<wref word="Sq" />
```

will look in the current word set context for the word with the id "Sq", looking for the label "⟨*wordset*⟩:Sq". If this is not found, it will look in the core word set (the label "core:Sq"). It will then output the following hyper link:

6.1.2165 S"

The function <rref> references the rationale for the word (prefixes the label with "`rat:`") effectively providing a separate label name space. The <tref> functions references the testing ("`test:`") name space while <iref> references the reference implementation ("`imp:`") name space.

Question:

This describes the operation of the existing LaTeX macros. The name space could be passed to the generate cross reference function:

⟨*cross-ref*⟩ := *<ref* ⟨*att-ref*⟩ [⟨*namespace=" (rat | test | imp) "*⟩] */>* | ⟨*xref*⟩

where the label is made up from "⟨namespace⟩:⟨wordset⟩:⟨word⟩".

An alternative would be to insist the user provides a full label.

Finally, the <xref> tag allows for the cross referencing between sections of the document. If the section identified by the label attribute can be found, a full reference to the section is produced, complete with a link to the section. However, if the section cannot be found the ⟨*text*⟩ is used.

Question:

The <xref> tag is already used by DocBook so we may need to use a different tag for this function. Suggestions as to the name?

2 Block / Paragraph text

The ⟨*sec-text*⟩ element consists of any number of block or paragraph level elements described by the ⟨*para-text*⟩ element.

⟨*para-text*⟩ := ⟨*stack*⟩ | ⟨*note*⟩ | ⟨*item*⟩ | ⟨*code*⟩ | ⟨*DBpara*⟩

⟨*type*⟩ := type="⟨*string*⟩"

⟨*note*⟩ := <note [⟨*type*⟩]> ⟨*text*⟩ </note>

⟨*item*⟩ := <item ⟨*type*⟩> ⟨*text*⟩ </item>

⟨*DBpara*⟩ := ... *DocBook paragraph level elements* ...

Some sections take an optional type attribute. This is normally displayed in the label for that section of the definition, with the exception of the <item> section where the type is the whole of the label.

The list of paragraph level DocBook elements includes the <para> (paragraph) element.

2.1 Stack description

Many sections will require a stack description. The <stack> element is considered a paragraph (block) level element:

⟨*stack*⟩ := <stack [type="⟨*upper*⟩"]>
[<pre> ⟨*stack-item*⟩* </pre>]
[<post> ⟨*stack-item*⟩* </post>]
</stack>

⟨*stack-item*⟩ := ⟨*char*⟩* | | | _⟨*char*⟩ | _{ ⟨*char*⟩* } | * | ⟨*parsed-text*⟩

⟨*parsed-text*⟩ := < (char | chars | space | spaces | quote | paren | eol) />

A stack description gives both the before (<pre>) and after (<post>) items. The type attribute is used to indicate the which stack the stack description is being being applied. (I.e., type="R" for the return stack).

The ⟨*char*⟩ element of the ⟨*stack-item*⟩ is rather misleading as a number of characters are treated as "special". The special meaning has been taken from the LaTeX macros:

`x_{i}`	x subscript i	x_i
`x_i`	x subscript i	x_i
`x*i`	x multiplied by i	$x * i$
`n|u`	n or u	$n\|u$
`"name<space/>"`	the syntax element *name* followed by a space	"*name*⟨*space*⟩"

The intention was to make the stack item appear as similar to the output text as possible.

Table 2.1 of the '94 document defines the seven syntactic elements (in ⟨*parsed-text*⟩) that can be used when parsing text. Defining these as empty elements allows there use in this way.

Question:

Should we consider using XML for the stack notation?

If so, then what notation, something along the lines of:

x_i ⇒ x_i
x<times/>y ⇒ $x * y$
n∣u ⇒ $n|u$
n❘u ⇒ $n|u$

3 Inline text

A number of additional functions are provided for text level. These functions can be used within any paragraph / block level element.

This is normal text with the following additional in-line tags:

⟨*text*⟩ := ⟨*DBinline*⟩ | ⟨*cross-ref*⟩ | ⟨*word*⟩ | ⟨*c*⟩
| <param> ⟨*stack item*⟩* </param>
| <arg> ⟨*string*⟩ </arg>

⟨*word*⟩ := <word ⟨*att-ref*⟩ />

The following additional formatting tags are provided:

⟨*DBinline*⟩	DocBook inline elements, including <emphasis>.
⟨*cross-ref*⟩	The cross reference functions (⟨*wref*⟩, ⟨*rref*⟩, ⟨*tref*⟩, ⟨*iref*⟩, ⟨*xref*⟩).
<word>	Represents a forth word. The word and wordset attributes work in the same way as for the cross reference tags. This will only output the name of the referenced word. If this is the current word, the name is output in bold, otherwise the word is a link to the word's definition. <word word="IF"/> ⇒ IF <word word="Sq" wordset="file" /> ⇒ S" <word word="DOES" /> ⇒ **DOES>**
⟨*c*⟩	Represents program code fragment within the body of the text.
<param>	Represents a stack item within the the text. <param>c-add len</param> ⇒ *c-addr len*
<arg>	Represents an argument <arg>text</arg> ⇒ ⟨*text*⟩

4 Source code

Source code is output in one of two environments: the paragraph level <code> tag; the inline text level <c> tag.

⟨*code*⟩ := <code> ⟨*multi-line-source*⟩ </code>

⟨*c*⟩ := <c> ⟨*inline-source*⟩ </c>

⟨*multi-line-source*⟩ := (⟨*para-text*⟩ | ⟨*inline-source*⟩)*

⟨*inline-source*⟩ := (⟨*char*⟩ | ⟨*word*⟩)*

In both environments the white space is preserved. All Forth words are cross referenced back to their definition using the <word> tag.

Question:

We could dispense with the <word> tag and assume every sequence of non-white spaces is a label for a word. Any label not found in the current word set or the core word set can be output directly.

The the current, rather user hostile, example:

```
<word word=":" /> TEST <word word="POSTPONE" /> <word word="Sq" wordset="file" />
...  <word word=";" />
```

would become the considerable more friendly:

```
: TEST POSTPONE file:Sq ...  ;
```

Note the use of the word's label rather than the word name, i.e., Sq and not S". Also note the use of the word set prefix.

While this would be considerable more user friendly, I have no idea how to go about implementing this is XSLT.