1. ATL Transformation Example

1.1. Example: XSLT → XQuery

The XSLT to XQuery example (originally described in [1]) describes a simplified transformation of XSLT code to XQuery code.

1.1.1. Metamodels

The source metamodel of XSLT has been modelled for this simplified transformation example. It is based on an XML metamodel, which it extends. Consequently, the XML metamodel has to be explained before the XSLT metamodel.

The XML metamodel presented (see Figure 1 XML) describes an XML document (Document) composed of one root node (RootNode). Node is an abstract class having two direct children, namely ElementNode and AttributeNode. ElementNode represents the tags, for example a tag named xml: <xml></xml>. ElementNodes can be composed of many Nodes. AttributeNode represents attributes, which can be found in a tag, for example the attr attribute: <xml attr="value of attr"/>. ElementNode has two sub classes, namely RootNode and TextNode. RootNode is the root element. The TextNode is a particular node, which does not look like a tag; it is only a string of characters.

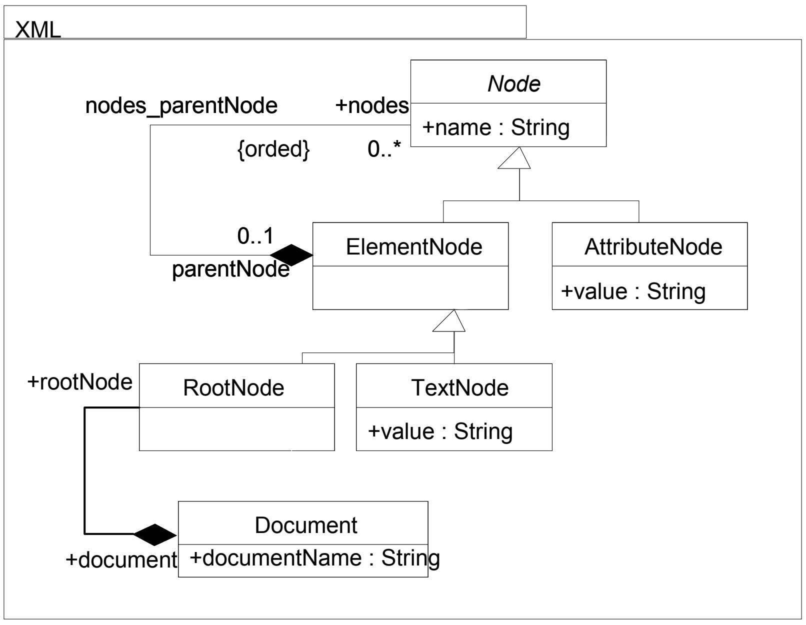

Figure 1 XML

The XSLT metamodel developed for this example (see Figure 2 XSLT) is an extension of the XML metamodel. The extension consists of classes represented in grey. The main class is called XSLTNode and inherits from ElementNode. The XSLTNode class has sub classes representing XSLT elements, namely xsl:apply-templates, xsl:template, xsl:if, xsl:value-of. For reasons of simplification, several features such as xsl:for-each, xsl:choose, xsl:sort, xsl:copy-of elements have been ignored; this is why these are neither in the metamodel nor in the transformation code.

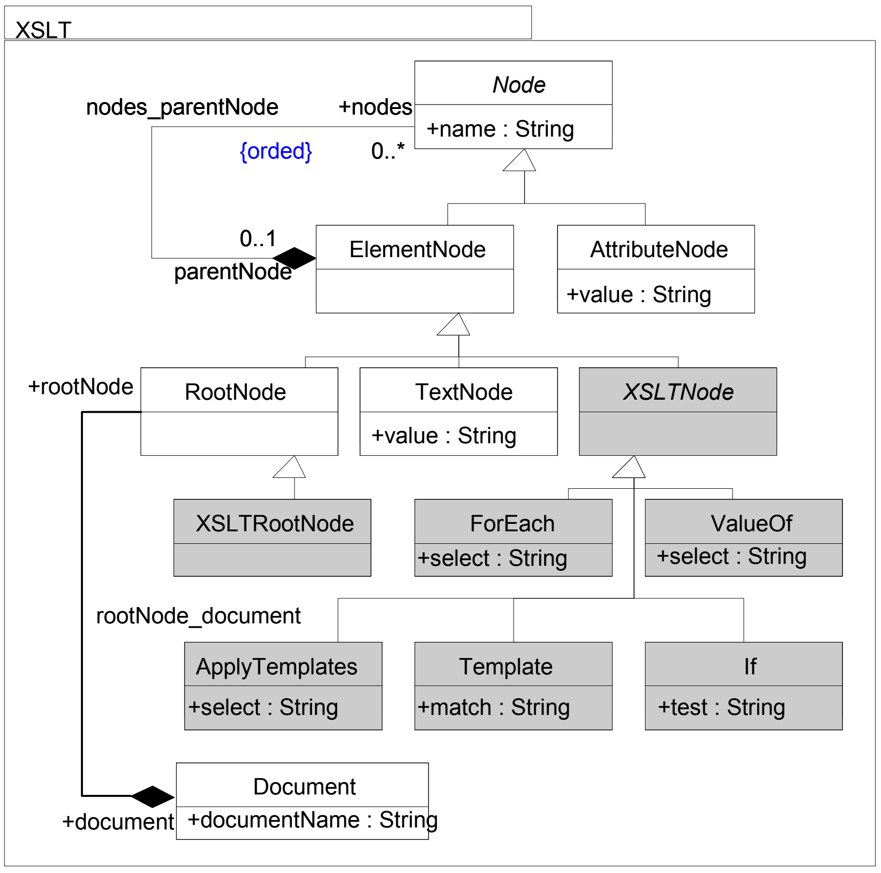

Figure 2 XSLT

The target metamodel of this example is XQuery (see Figure 3 XQuery). It contains also parts of the XML metamodel (Node, ElementNode, AttributeNode and TextNode). An XQueryProgram is composed of ExecutableExpressions which can be FLWOR expressions, function calls (FunctionCall) and function declarations (FunctionDeclaration).

The main class is FLWOR, it represents FLWOR expressions which are composed of For, Let, Where, Order by and Return statements. For is composed of an XPath expression representing the value stored by the variable defined by the var attribute. Let is also composed of an XPath expression representing the value stored by the variable defined by the var attribute. Where is composed of a boolean XPath expression used to do a selection on the variables of the For statements. OrderBy is composed of an XPath expression defining how to order the output. Return is composed of Expressions representing the output data. Expression is the superclass of ExecutableExpressions, (XML-) Nodes and ReturnXPath expressions. The Node class and its sub classes are the same as those of the XML metamodel. There are two different XPath classes. In the ReturnXPath class the corresponding String expression (value) has to be put between braces, in the XPath class the expression is without braces.

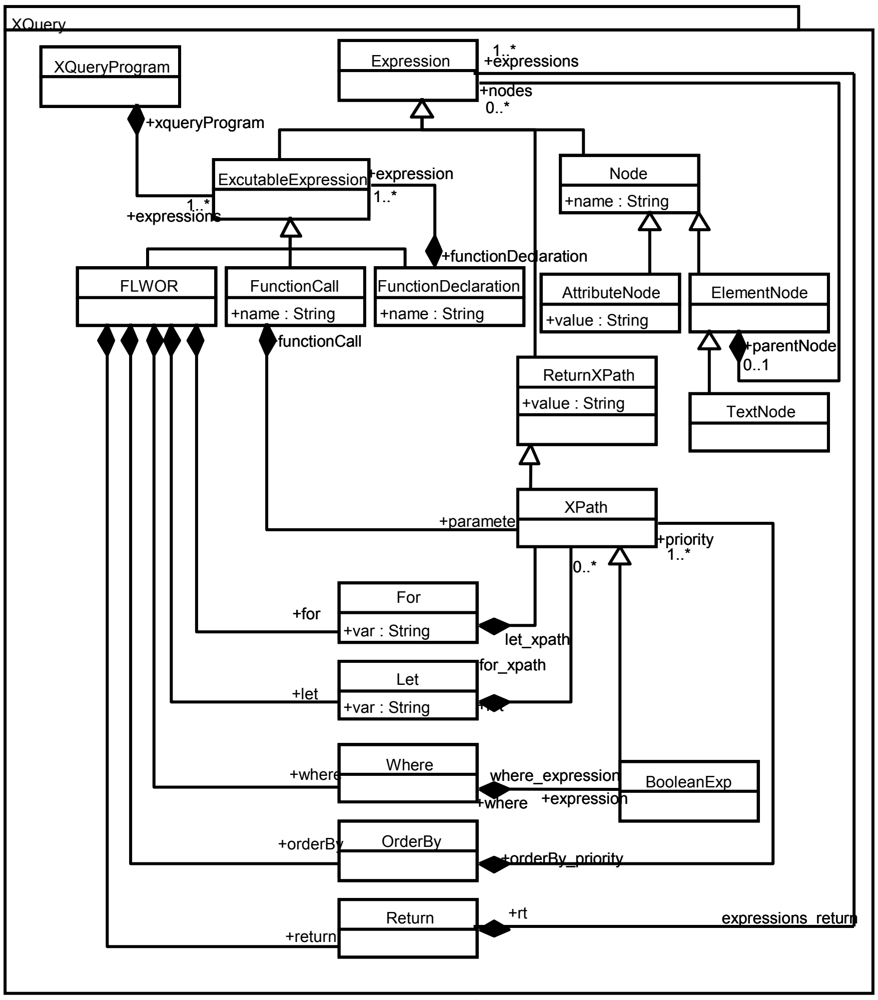

Figure 3 XQuery

1.1.2. Rules Specification

The transformation can be divided into three types of rules:

- the rule for the creation of an XQueryProgram from an XSLTRootNode instance,
- the rules for the transformation of XSLT elements into XQuery expressions and
- the rules for copying XML elements and XML attributes.

These are the rules to transform an XSLT model to an XQuery model:

- For the XSLTRootNode instance, an XQueryProgram instance has to be created. This involves follow-up instantiations and value attributions:
  - A new FLWOR instance has to be created and its references have to be set.
    - The xQueryProgram reference has to be set to the newly created XQueryProgram. The for and the return references have to point to the corresponding instances that will be described in the following.

  - A new For instance has to be created.
    - Its var attribute has to be set to '$var'.
    - Its expression reference has to point to the XPath instance that will be described in the following.
  - A new XPath instance has to be created.
    - Its value is set to 'document(\"xmlFile.xml\")'
  - A new Return instance has to be created.
    - The expressions reference set in Return has to contain all those grandchildren nodes (defined by the recursive use of the nodes reference in ElementNode) of which the children Template nodes of the XSLTRootNode have the match value '/'. In other words, select all instances of XSLTRootNode referenced in nodes and choose all those Templates having the match value '/'. Let the expressions references of this Return instance point to all those elements referenced by the elements that correspond to the nodes of the chosen Templates.

- For each XSLT Template instance, an XQuery FunctionDeclaration instance has to be created, if the match value is not '/'. This involves follow-up instantiations.
  - The new FunctionDeclaration instance has the following value and references:
    - The name of the FunctionDeclaration is 'fct' concatenated with the match String.
    - Its expression reference is a sequence of FLWOR instances that will be described below.
    - Its xQueryProgram reference points to the first XSLTRootNode instance.
  - A new FLWOR has to be created.
    - Its for and return references have to point to the corresponding instances described in the following.
  - A new For instance has to be created.
    - Its var value is '$var'.
    - Its forExpresson points to the corresponding XPath instance described in the following.
  - A new Path instance has to be created.
    - Its value attribute has to be set to '$paramVar'
  - A new Return instance has to be created.
    - Its expressions references corresponds the node references of Template.
- For each XSLT If instance, an XQuery FLWOR instance has to be created.

- This involves also the instantiation of a Let, a Where and a Return variable which have to be referenced by the corresponding references (let, where and return) in this FLWOR instance.
- A new Let instance has to be created.
  - The expression reference has to reference the new XPath instance described below.
  - The var attribute has to be set to ‘$var’.
- A new XPath instance has to be created.
  - The value attribute has to be set to ‘$var’.
- A new Where instance has to be created.
  - The expression reference has to reference the new BooleanExp instance described below.
- A new BooleanExp has to be created.
  - Its value attribute has to be set to ‘$var’ concatenated with the test attribute of the If instance.
- A new Return instance has to be created.
  - Its expressions references have to point to the elements that correspond to the nodes references of the If instance.

- For each XSLT ApplyTemplate instance, an XQuery FunctionCall instance has to be created.
    - The name attribute has to be set to ‘fct’ concatenated with the select attribute of the ApplyTemplate instance.
    - Its parameters reference has to reference the XPath described below.
  - A new XPath instance has to be created.
    - Its value attribute has to be set to ‘$var’ concatenated with the select attribute of the ApplyTemplate instance.
- For each XSLT ValueOf instance, an XQuery ReturnXPath instance has to be created.
  - Its value attribute has to be set to ‘$var’ concatenated with the _valueOf attribute of the ValueOf instance.
- For each XSLT ElementNode instance that has a name different from xsl:otherwise, xsl:when, xsl:choose, xsl:copy-of, xsl:sort, xsl:foreach, xsl:if, xsl:apply-template, xsl:value-of, xsl:template and xsl:stylesheet, an XQuery ReturnXPath instance has to be created.
  - The name attributes of ElementNode and ReturnXPath have to correspond.
  - The nodes references of ElementNode and ReturnXPath have to correspond.
- For each XSLT AttributeNode instance, an XQuery AttributeNode instance has to be created with the same values.
  - The name attributes of ElementNode and ReturnXPath have to correspond.

- The value attributes of ElementNode and ReturnXPath have to correspond.

The transformation described is simplified with the following constraints:

- All the template tags must be direct children of the root node. This constraint simplifies the behaviour of templates.
- The value of a select attribute of an apply-template must be a tag name, it must not be an XPath expression. This constraint hides the main difference between a template and a function call. An apply-template tag applies all available templates to a set of elements and each template treats only the elements that it is dedicated to. Whereas a function call applies a function to a set of elements; the test of the type of the elements must be explicitly described in the function declaration.
- The XSLT programmer has to write one template matching to '/'. It defines indirectly the starting point. This information is necessary with respect to the XQuery program. XQuery is partly an imperative language; it defines the order of the program execution.

References

[1] Bézivin, J., Dupé, G., Jouault, F., Pitette, G., Rougui, EJ. First experiments with the ATL model transformation language: Transforming XSLT into XQuery. OOPSLA 2003 Workshop, California, 2003.