Vacation Days Challenge

Bruce Silver (bruce@brsilver.com)

As I understand it, the challenge is to ingest the XML file VacationDaysChallenge.xml (Figure 1) and execute it for some test data.

```
<?altova_sps dmn2.sps?>
<definitions xmlns="http://www.omg.org/spec/DMN/20151101/dmn.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:feel="http://www.omg.org/spec/FEEL/20140401"
name="DMChallenge" namespace="methodandstyle/dmchallenge"
xsi:schemaLocation="http://www.omg.org/spec/DMN/20151101/dmn.xsd dmn.xsd" exporter="methodandstyle"
exporterVersion="1.0" slick-uniqueid="3">
  <div>
    <a id="slick_uniqueid"/>
  </div>
  <inputData name="Age" id="i_Age">
    <variable name="Age" typeRef="feel:number"/>
  </inputData>
  <inputData name="YearsOfService" id="i_YearsOfService">
    <variable name="YearsOfService" typeRef="feel:number"/>
  </inputData>
  <decision name="VacationDays" id="d_VacationDays">
    <variable name="VacationDays" typeRef="feel:number"/>
    <informationRequirement>
      <requiredInput href="#i_Age"/>
    </informationRequirement>
    <informationRequirement>
      <requiredInput href="#i_YearsOfService"/>
    </informationRequirement>
    <decisionTable>
      <input>
        <inputExpression>
          <text>Age</text>
        </inputExpression>
      </input>
      <input>
        <inputExpression>
          <text>YearsOfService</text>
        </inputExpression>
      </input>
      <output name="VacationDays"/>
      <rule>
        <inputEntry>
          <text><18</text>
        </inputEntry>
        <inputEntry>
          <text>-</text>
        </inputEntry>
        <outputEntry>
          <text>22+5</text>
        </outputEntry>
      </rule>
      <rule>
        <inputEntry>
          <text>[18..45)</text>
        </inputEntry>
        <inputEntry>
```

Figure 1. VacationDaysChallenge.xml on DM challenge website

My solution (Figure 2) is based on XSLT 2.0, chosen because FEEL is essentially a subset of XPATH 2.0, so mapping FEEL built-in functions is not so hard. This particular challenge doesn't use FEEL functions.

It works in 2 steps. The transform libGen.xslt maps FEEL/S-FEEL expressions from the model VacationDaysChallenge.xml to equivalent XPATH, saved in another xslt transform, VacationDaysChallenge-Functions.xslt. This file is a function library that is referenced by include in the second transform dmnExecute.xslt. dmnExecute.xslt transforms the instance document dmChallenge-instance.xml to produce the output, a listing of all decision nodes in the model and their values.

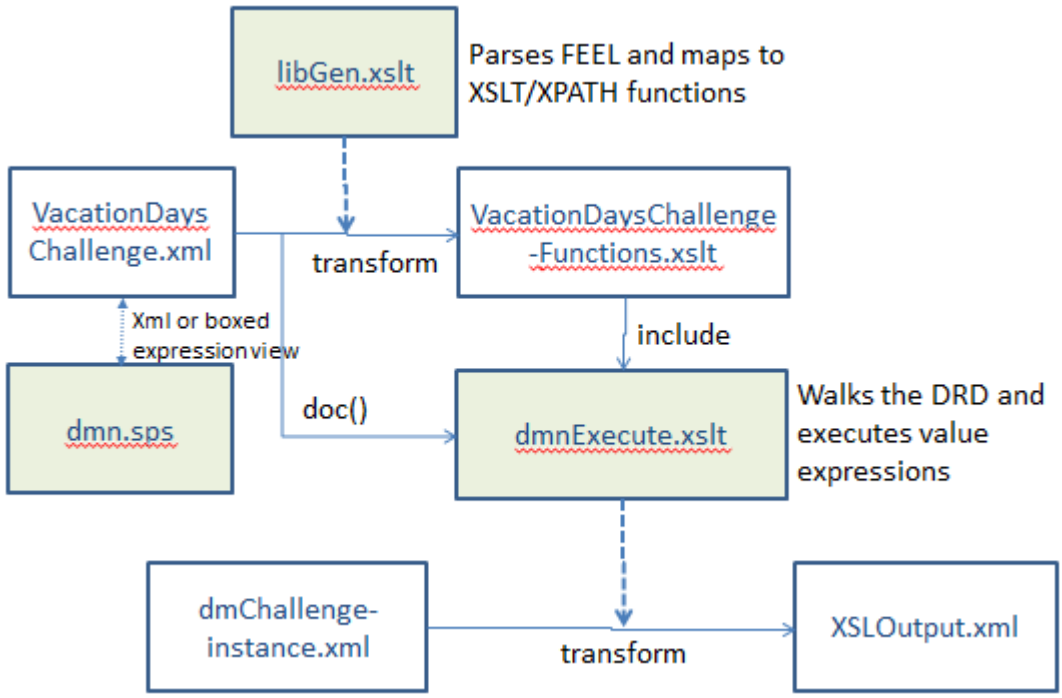


Figure 2. Solution architecture

A nice feature of XML is that you do not need to view it as text markup, but you can view it in user-defined layouts using xslt as well. Altova Stylevision lets you create html forms to view and directly edit the xml, called Authentic views accessible in XMLSpy. Figure 3 shows the model in XMLSpy text view and Figure 4 shows what it looks like in Authentic view: as boxed expressions. The Authentic view makes it convenient to create DMN model XML directly via boxed expressions.

Figure 5 is a snippet of the function library VacationDaysChallenge-Functions.xslt created by transforming the model. This file is specific to a decision model but independent of the instance values.

Figure 6 is a snippet of the dmnExecute.xslt, which walks the DRD – trivial in this particular case – and executes the decision logic for each decision node and BKM node. Note line 3 includes the model-specific function library. In this version, one must hard-code the name of the function library in line 3, but it could be passed as a global parameter.

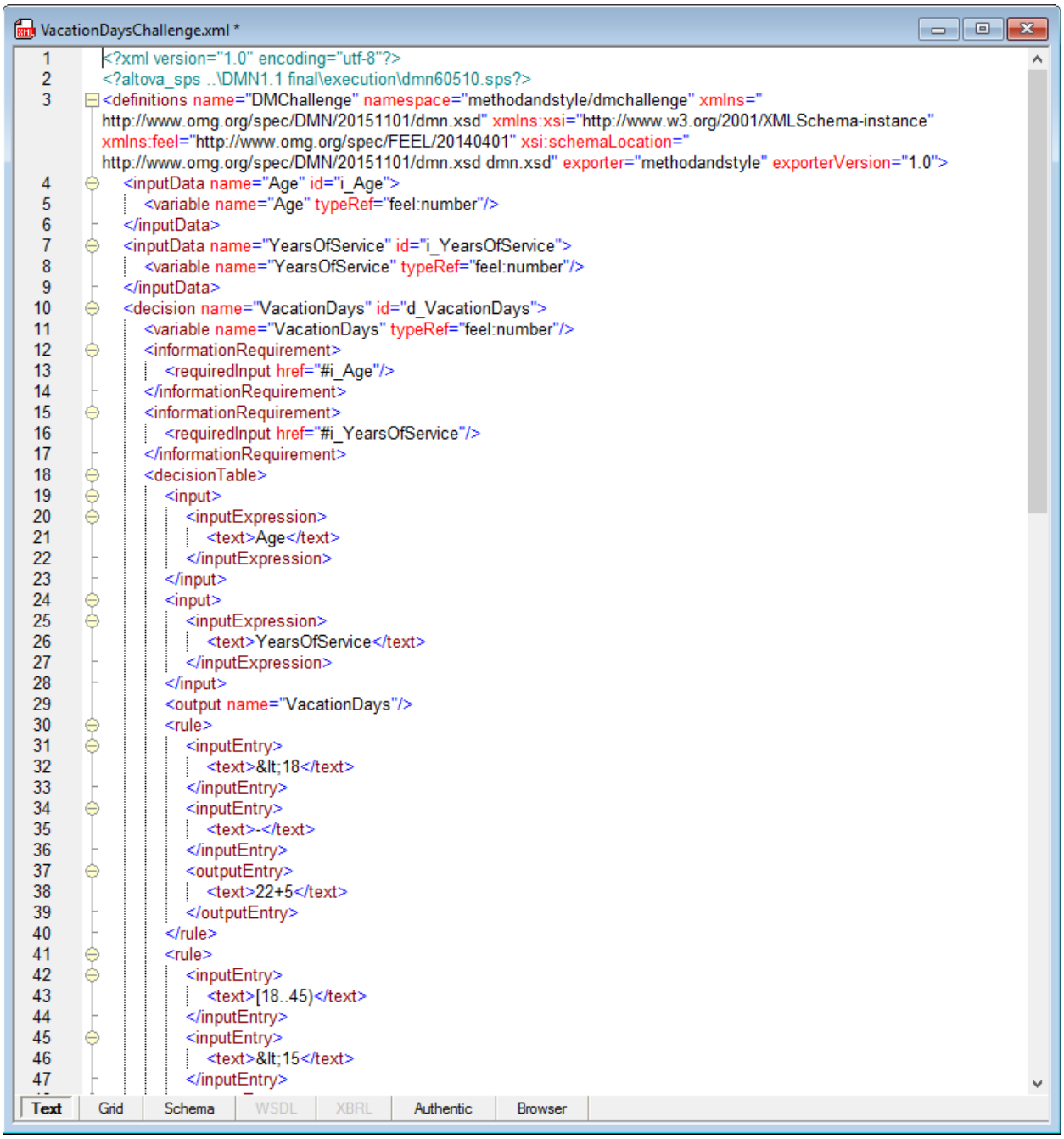
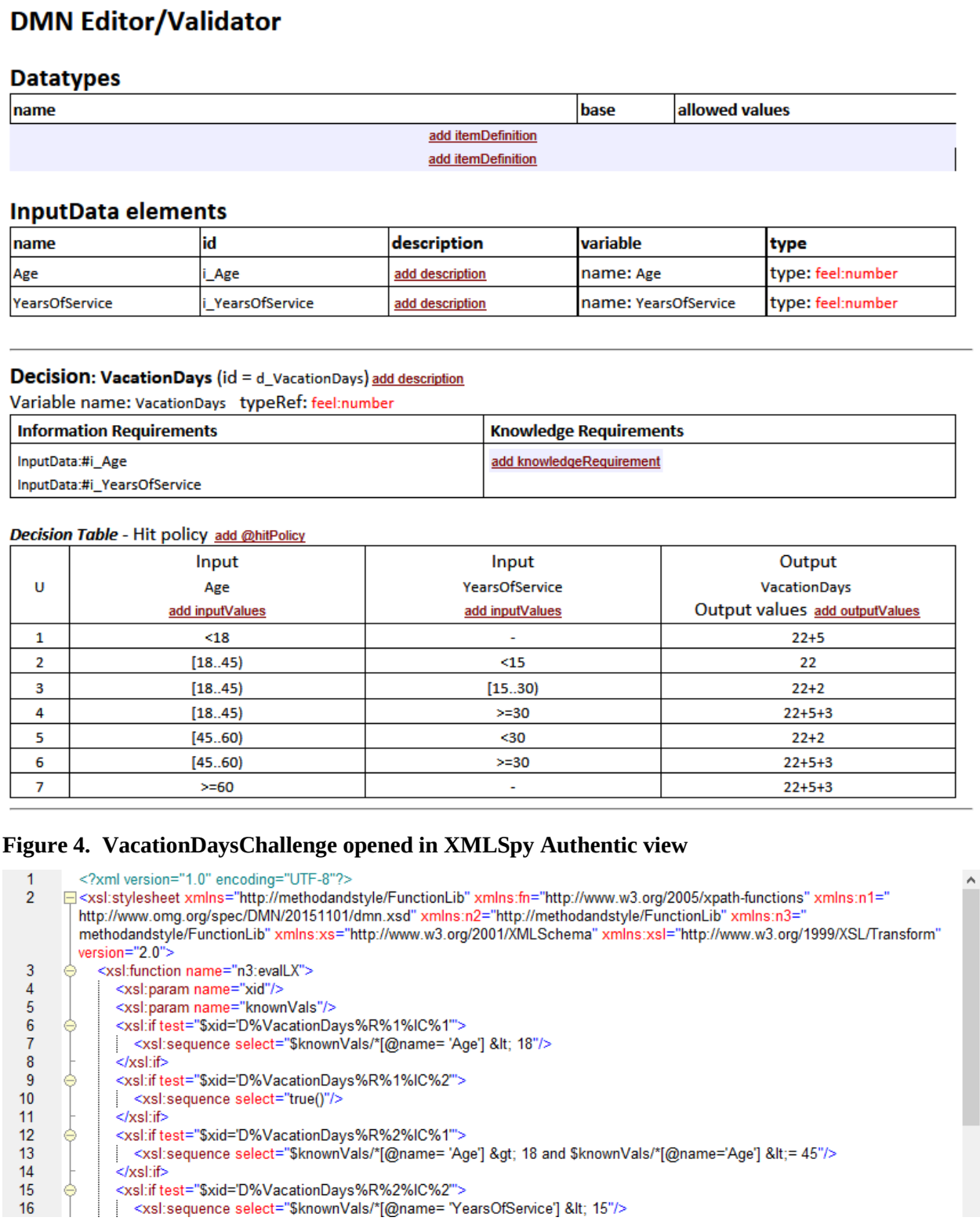


Figure 3. VacationDaysChallenge opened in XMLSpy Text view



DMN Editor/Validator

Datatypes

name	base	allowed values
add itemDefinition		
add itemDefinition		

InputData elements

name	id	description	variable	type
Age	i_Age	add description	name: Age	type: feel:number
YearsOfService	i_YearsOfService	add description	name: YearsOfService	type: feel:number

Decision: VacationDays (id = d_VacationDays) add description

Variable name: VacationDays typeRef: feel:number

Information Requirements	Knowledge Requirements
InputData:#i_Age InputData:#i_YearsOfService	add knowledgeRequirement

Decision Table - Hit policy add @hitPolicy

U	Input Age add inputValues	Input YearsOfService add inputValues	Output VacationDays Output values add outputValues
1	<18	-	22+5
2	[18..45)	<15	22
3	[18..45)	[15..30)	22+2
4	[18..45)	>=30	22+5+3
5	[45..60)	<30	22+2
6	[45..60)	>=30	22+5+3
7	>=60	-	22+5+3

Figure 4. VacationDaysChallenge opened in XMLSpy Authentic view

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <xsl:stylesheet xmlns="http://methodandstyle/FunctionLib" xmlns:fn="http://www.w3.org/2005/xpath-functions" xmlns:n1="
    http://www.omg.org/spec/DMN/20151101/dmn.xsd" xmlns:n2="http://methodandstyle/FunctionLib" xmlns:n3="
    methodandstyle/FunctionLib" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="2.0">
3     <xsl:function name="n3:evalLX">
4       <xsl:param name="xid"/>
5       <xsl:param name="knownVals"/>
6       <xsl:if test="$xid='D%VacationDays%R%1%IC%1'">
7         <xsl:sequence select="$knownVals/*[@name= 'Age'] &lt; 18"/>
8       </xsl:if>
9       <xsl:if test="$xid='D%VacationDays%R%1%IC%2'">
10        <xsl:sequence select="true()"/>
11      </xsl:if>
12      <xsl:if test="$xid='D%VacationDays%R%2%IC%1'">
13        <xsl:sequence select="$knownVals/*[@name= 'Age'] &gt; 18 and $knownVals/*[@name='Age'] &lt;= 45"/>
14      </xsl:if>
15      <xsl:if test="$xid='D%VacationDays%R%2%IC%2'">
16        <xsl:sequence select="$knownVals/*[@name= 'YearsOfService'] &lt; 15"/>
```

Figure 5. Snippet of VacationDaysChallenge-Functions.xslt created by transforming the model using libGen.xslt

dmnexecute-new.xslt

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:fn="
    http://www.w3.org/2005/xpath-functions" xmlns:n1="
    http://www.omg.org/spec/DMN/20151101/dmn.xsd" xmlns="methodandstyle/dmnExecute"
     xmlns:tns="methodandstyle/dmnExecute" xmlns:n2="methodandstyle/dmnInstance"
    xmlns:n3="methodandstyle/FunctionLib">
3       <xsl:include href="VacationDaysChallenge-Functions.xslt"/>
4       <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
5       <xsl:param name="dmn" select="doc(//n2:decisionModelName)"/>
6       <xsl:template match="/">
7           <xsl:variable name="instance" select="//n2:inputData|//n2:decision"/>
8           <!--$kList is a list of nodes with known value-->
9           <xsl:variable name="kList" as="item()*">
10              <kList xmlns="methodandstyle/dmnInstance">
11                  <xsl:for-each select="$instance">
12                      <xsl:variable name="theNode">
13                          <node name="{@name}" type="inputData">
14                              <xsl:choose>
15                                  <xsl:when test="*">
16                                      <xsl:copy-of select="*"/>
17                                  </xsl:when>
18                                  <xsl:otherwise>
19                                      <xsl:value-of select="."/>
20                                  </xsl:otherwise>
21                              </xsl:choose>
22                          </node>
23                      </xsl:variable>
24                      <xsl:sequence select="$theNode"/>
25                  </xsl:for-each>
26              </kList>
27          </xsl:variable>
```

Figure 6. Snippet of dmnExecute.xslt, which operates on an xml instance file

dmChallenge-instance.xml

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <?xml-stylesheet type="text/xsl" href="dmnExecute-new.xslt"?>
3   <decisionModelInstance xmlns="methodandstyle/dmnInstance">
4       <decisionModelName>VacationDaysChallenge.xml</decisionModelName>
5       <inputData name="Age">48</inputData>
6       <inputData name="YearsOfService">30</inputData>
7   </decisionModelInstance>
8
```

Figure 7. The instance file dmnChallenge-instance.xml

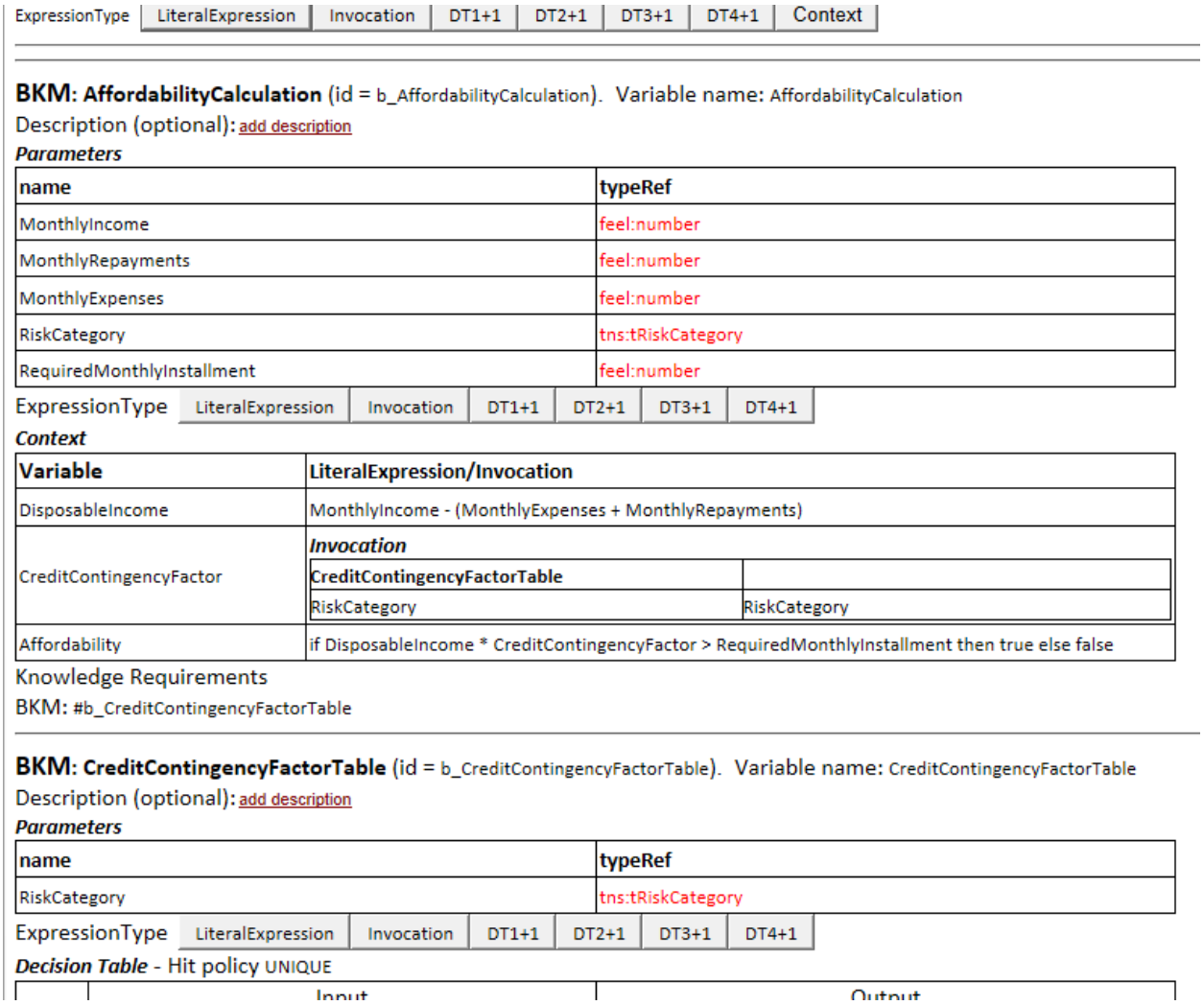
XSL Output.xml

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <kList xmlns="methodandstyle/dmnInstance" xmlns:fn="http://www.w3.org/2005/xpath-functions"
     xmlns:n1="http://www.omg.org/spec/DMN/20151101/dmn.xsd" xmlns:n2="
    methodandstyle/dmnInstance" xmlns:n3="methodandstyle/FunctionLib" xmlns:tns="
    methodandstyle/dmnExecute" xmlns:xs="http://www.w3.org/2001/XMLSchema">
3       <node name="Age" type="inputData">48</node>
4       <node name="YearsOfService" type="inputData">30</node>
5       <node name="VacationDays" type="decision">30</node>
6   </kList>
```

Figure 8. The output generated by dmnExecute.xslt, a list of all DRD nodes and their values

As an example of FEEL/boxed expression support, this example is too trivial to be very interesting. A better illustration might be the Lending decision example in the DMN spec. It has a more complex DRD, including invocations, contexts (including a nested invocation), if..then..else operators, the amortization formula (including exponentiation, not a built-in in XPATH 2.0) and other more demanding stuff. Except for removing spaces from the names of DRG elements, the model is exactly the same as presented in Chapter 11 of the spec.

Figure 9 is a snippet of the Authentic view of this model. Figure 10 is a snippet of the function library. (Note line 8 is the amortization formula mapped to XPATH.) Figure 11 is the instance file, and Figure 12 is the output generated from dmnExecute.xslt.



ExpressionType | LiteralExpression | Invocation | DT1+1 | DT2+1 | DT3+1 | DT4+1 | Context

BKM: AffordabilityCalculation (id = b_AffordabilityCalculation). Variable name: AffordabilityCalculation

Description (optional): add description

Parameters

name	typeRef
MonthlyIncome	feel:number
MonthlyRepayments	feel:number
MonthlyExpenses	feel:number
RiskCategory	tns:tRiskCategory
RequiredMonthlyInstallment	feel:number

ExpressionType | LiteralExpression | Invocation | DT1+1 | DT2+1 | DT3+1 | DT4+1

Context

Variable	LiteralExpression/Invocation	
DisposableIncome	MonthlyIncome - (MonthlyExpenses + MonthlyRepayments)	
CreditContingencyFactor	***Invocation***	
	CreditContingencyFactorTable	
	RiskCategory	RiskCategory
Affordability	if DisposableIncome * CreditContingencyFactor > RequiredMonthlyInstallment then true else false	

Knowledge Requirements

BKM: #b_CreditContingencyFactorTable

BKM: CreditContingencyFactorTable (id = b_CreditContingencyFactorTable). Variable name: CreditContingencyFactorTable

Description (optional): add description

Parameters

name	typeRef
RiskCategory	tns:tRiskCategory

ExpressionType | LiteralExpression | Invocation | DT1+1 | DT2+1 | DT3+1 | DT4+1

Decision Table - Hit policy UNIQUE

	Input	Output

Figure 9. Snippet of Lending example decision model in Authentic view

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns="http://methodandstyle/FunctionLib" xmlns:fn="http://www.w3.org/2005/xpath-functions" xmlns:n1="http://www.omg.org/spec/DMN/20151101/dmn.xsd"
xmlns:n2="http://methodandstyle/FunctionLib" xmlns:n3="methodandstyle/FunctionLib" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsl="
http://www.w3.org/1999/XSL/Transform" version="2.0">
  <xsl:function name="n3:evalLX">
    <xsl:param name="xid"/>
    <xsl:param name="knownVals"/>
    <xsl:choose>
      <xsl:when test="$xid='C%InstallmentCalculation.MonthlyRepayment'">
        <xsl:sequence select="( $knownVals/*[@name='Amount'] * $knownVals/*[@name='Rate'] div 12 ) div ( 1 - n3:pow( ( 1 + $knownVals/*[@name='Rate'] div 12 ) , -
$knownVals/*[@name='Term'] ) )"/>
      </xsl:when>
      <xsl:when test="$xid='D%Strategy%R%1%IC%1'">
        <xsl:sequence select="$knownVals/*[@name= 'Eligibility']='INELIGIBLE'"/>
      </xsl:when>
      <xsl:when test="$xid='D%Strategy%R%1%IC%2'">
        <xsl:sequence select="true()"/>
      </xsl:when>
      <xsl:when test="$xid='D%Strategy%R%2%IC%1'">
        <xsl:sequence select="$knownVals/*[@name= 'Eligibility']='ELIGIBLE'"/>
      </xsl:when>
      <xsl:when test="$xid='D%Strategy%R%2%IC%2'">
        <xsl:sequence select="$knownVals/*[@name= 'BureauCallType']='FULL' or $knownVals/*[@name= 'BureauCallType']='MINI'"/>
      </xsl:when>
      <xsl:when test="$xid='D%Strategy%R%3%IC%1'">
        <xsl:sequence select="$knownVals/*[@name= 'Eligibility']='ELIGIBLE'"/>
      </xsl:when>
      <xsl:when test="$xid='D%Strategy%R%3%IC%2'">
        <xsl:sequence select="$knownVals/*[@name= 'BureauCallType']='NONE'"/>
      </xsl:when>
      <xsl:when test="$xid='B%CreditContingencyFactorTable%R%1%IC%1'">
        <xsl:sequence select="$knownVals/*[@name= 'RiskCategory']='HIGH' or $knownVals/*[@name= 'RiskCategory']='DECLINE'"/>
```

Figure 10. Snippet of function library for Lending example

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="dmnExecute-new.xslt"?>
<decisionModelInstance xmlns="methodandstyle/dmnInstance">
  <decisionModelName>Lending-original.xml</decisionModelName>
  <inputData name="ApplicantData">
    <Monthly>
      <Income>6000</Income>
      <Expenses>2000</Expenses>
      <Repayments>0</Repayments>
    </Monthly>
    <Age>35</Age>
    <ExistingCustomer>true</ExistingCustomer>
    <MaritalStatus>M</MaritalStatus>
    <EmploymentStatus>EMPLOYED</EmploymentStatus>
  </inputData>
  <inputData name="RequestedProduct">
    <ProductType>STANDARD LOAN</ProductType>
    <Amount>350000</Amount>
    <Rate>0.0395</Rate>
    <Term>360</Term>
  </inputData>
  <inputData name="BureauData">
    <CreditScore>649</CreditScore>
    <Bankrupt>false</Bankrupt>
  </inputData>
</decisionModelInstance>
```

Figure 11. Lending example, instance file

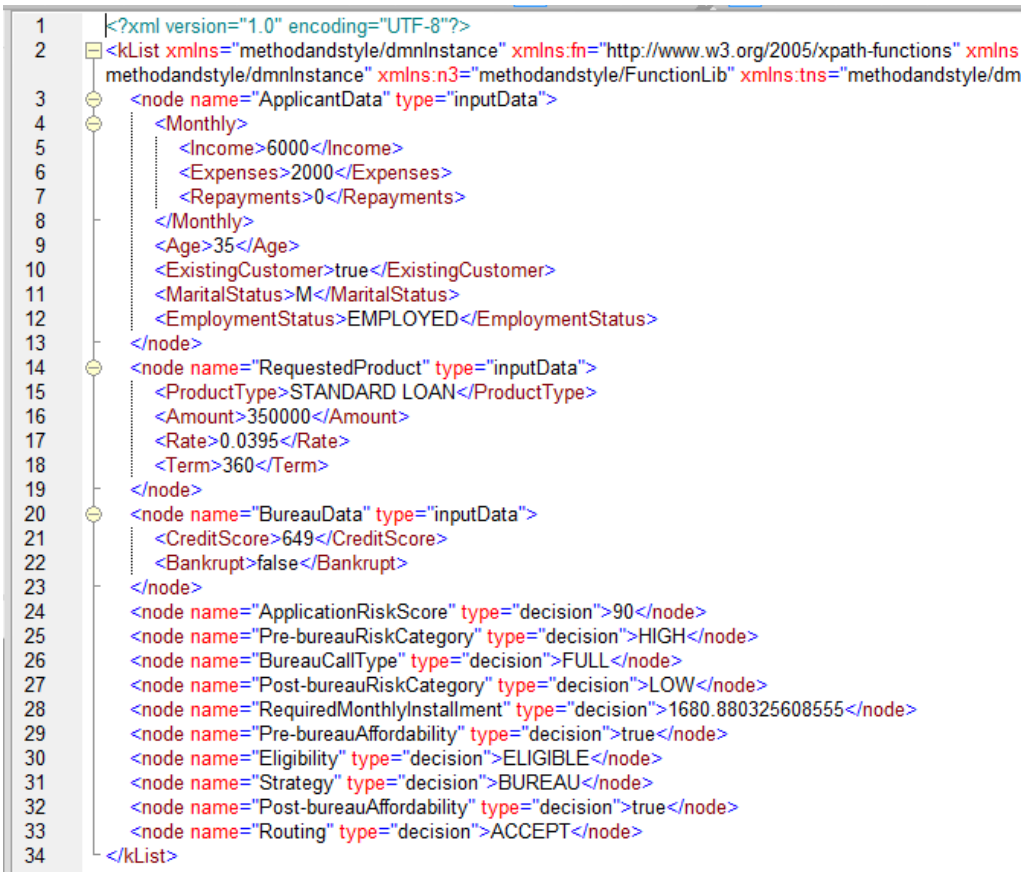


Figure 12. Lending example, output