Subject: Re: XmlNode class is excellent
Posted by koldo on Fri, 17 Oct 2014 23:41:21 GMT
View Forum Message <> Reply to Message

XML file and code to begin parsing it:

```
String xml = LoadFile(fileName);

try {
 XmlNode xn = ParseXML(xml);

 const XmlNode &nodeVariables = xn["dae"]["variables"];
 const XmlNode &nodeOrderedVariables = nodeVariables["orderedVariables"]["variablesList"];

 for (int i = 0; i < nodeOrderedVariables.GetCount(); ++i)
  Cout() << "\n " << nodeOrderedVariables[i].Attr("name");
...
```

Xml sample:

```
<?xml version="1.0" encoding="UTF-8"?>
<dae xmlns:p1="http://www.w3.org/1998/Math/MathML"
                                    xmlns:xlink="http://www.w3.org/1999/xlink"
                                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:noNamespaceSchemaLocation="http://home.dei.polimi.it/donida/Projects/AutoEdit/Images/DAE.xsd">
<variables dimension="7">
<orderedVariables dimension="5">
<variablesList>
<variable id="1" name="v_new" variability="discrete" direction="none" type="Real" fixed="false" flow="NonConnector" stream="NonStreamConnector">
<classesNames>
<element>
BouncingBall
</element>
<element>
Real
</element>
</classesNames>
</variable>
<variable id="2" name="impact" variability="discrete" direction="none" type="Boolean" fixed="false" flow="NonConnector" stream="NonStreamConnector">
<classesNames>
<element>
BouncingBall
</element>
<element>
Boolean
```

</element>
</classesNames>
</variable>
<variable id="3" name="flying" variability="discrete" direction="none" type="Boolean" fixed="false" flow="NonConnector" stream="NonStreamConnector" comment="true, if ball is flying">
<classesNames>
<element>
BouncingBall
</element>
<element>
Boolean
</element>
</classesNames>
<attributesValues>
<initialValue string="true">
</initialValue>
</attributesValues>
</variable>
<variable id="4" name="v" variability="continuousState" direction="none" type="Real" differentiatedIndex="1" fixed="false" flow="NonConnector" stream="NonStreamConnector" comment="velocity of ball">
<classesNames>
<element>
BouncingBall
</element>
<element>
Real
</element>
</classesNames>
</variable>
<variable id="5" name="h" variability="continuousState" direction="none" type="Real" differentiatedIndex="1" derivativeName="v" fixed="false" flow="NonConnector" stream="NonStreamConnector" comment="height of ball">
<classesNames>
<element>
BouncingBall
</element>
<element>
Real
</element>
</classesNames>
<attributesValues>
<initialValue string="5.0">
</initialValue>
</attributesValues>
</variable>
</variablesList>
</orderedVariables>

<knownVariables dimension="2">
<variablesList>
<variable id="1" name="g" variability="parameter" direction="none" type="Real" fixed="true"
flow="NonConnector" stream="NonStreamConnector" comment="gravity acceleration">
<bindValueExpression>
<bindExpression string="9.81">
</bindExpression>
</bindValueExpression>
<classesNames>
<element>
BouncingBall
</element>
<element>
Real
</element>
</classesNames>
</variable>
<variable id="2" name="e" variability="parameter" direction="none" type="Real" fixed="true"
flow="NonConnector" stream="NonStreamConnector" comment="coefficient of restitution">
<bindValueExpression>
<bindExpression string="0.8">
</bindExpression>
</bindValueExpression>
<classesNames>
<element>
BouncingBall
</element>
<element>
Real
</element>
</classesNames>
</variable>
</variablesList>
</knownVariables>
</variables>
<equations dimension="5">
<equation id="1">impact = h <= 0.0
</equation>
<equation id="2">der(v) = if flying then -g else 0.0
</equation>
<equation id="3">der(h) = v
</equation>
<whenEquation id="4">v_new := if impact and not pre(impact) then (-e) * pre(v) else 0.0
<whenEquationCondition>
{impact and v <= 0.0, impact}
</whenEquationCondition>
</whenEquation>
<whenEquation id="5">flying := v_new > 0.0

<whenEquationCondition>
{impact and v <= 0.0, impact}
</whenEquationCondition>
</whenEquation>
</equations>
<WhenClausesList dimension="1">
<WhenClause index="1">
<whenEquationCondition>
{impact and v <= 0.0, impact}
</whenEquationCondition>
<WhenOperatorsList dimension="1">
<WhenOperator>
reinit(v, v_new)
</WhenOperator>
</WhenOperatorsList>
</WhenClause>
</WhenClausesList>
<zeroCrossingList dimension="3">
<zeroCrossingElement string="h <= 0.0">
<involvedEquations>
<equationId>
1
</equationId>
</involvedEquations>
</zeroCrossingElement>
<zeroCrossingElement string="impact and v <= 0.0">
<involvedEquations>
<equationId>
4
</equationId>
</involvedEquations>
</zeroCrossingElement>
<zeroCrossingElement string="v <= 0.0">
<involvedEquations>
<equationId>
5
</equationId>
</involvedEquations>
</zeroCrossingElement>
</zeroCrossingList>
<functions>
</functions>
</dae>