NISTIR 7728

# Constrain Generator User's Guide

Salifou Sidi Malick
Matthew Molek
KC Morris

National Institute of Standards and Technology
U.S. Department of Commerce

**NISTIR 7728**

# Constrain Generator User's Guide

Salifou Sidi Malick
Matthew Molek
KC Morris
*Manufacturing Systems Integration Division*
*Manufacturing Engineering Laboratory*

August 2010

## Abstract

A common approach to systems integration is data exchange based on the XML series of standards. In this approach specifications for the data to be exchange are written in XML Schema. These specifications are written so as to be general enough that a large number of systems and partners can use them. Often the specifications are customized for the data exchanged in a particular scenario. One approach to customization is to write additional constraints for the data. This paper describes a tool which will assist a knowledgeable user in defining the customizations needed. The tool's interface is designed to guide the user through the data specification to add constraints on the specific data elements. The tool then turns those constraints into code which can be used to validate data for the exchange and generate a report of the results.

## Table of Contents

## 1) Introduction

This document describes the Constraint Generator. The Constraint Generator is a tool developed at the National Institute of Standards and Technology (NIST) to automate the generation of Schematron [1] scripts for an advance validation of XML [2] documents. The tool allows the user to open an XML Schema file [3] (i.e. an .xsd file), view a tree of the contents, and add constraints on the nodes by selecting them from the tree view. In this document we will describe the system requirements of the tool, then we will briefly describe how to use the tool including the user interfaces, and finally show an example of some of the tools output (script and report) based on a specific input schema.

## 2) System environment and Installation

The Constraint Generator is a cross-platform application developed and tested with Java 6.0 [4]. To install the tool, download the software from http://www.nist.gov/msid/XML_Testbed [5], extract the contents on a  hard drive, and read the README file for instructions on how to configure and launch the tool.

## 3) User Interfaces

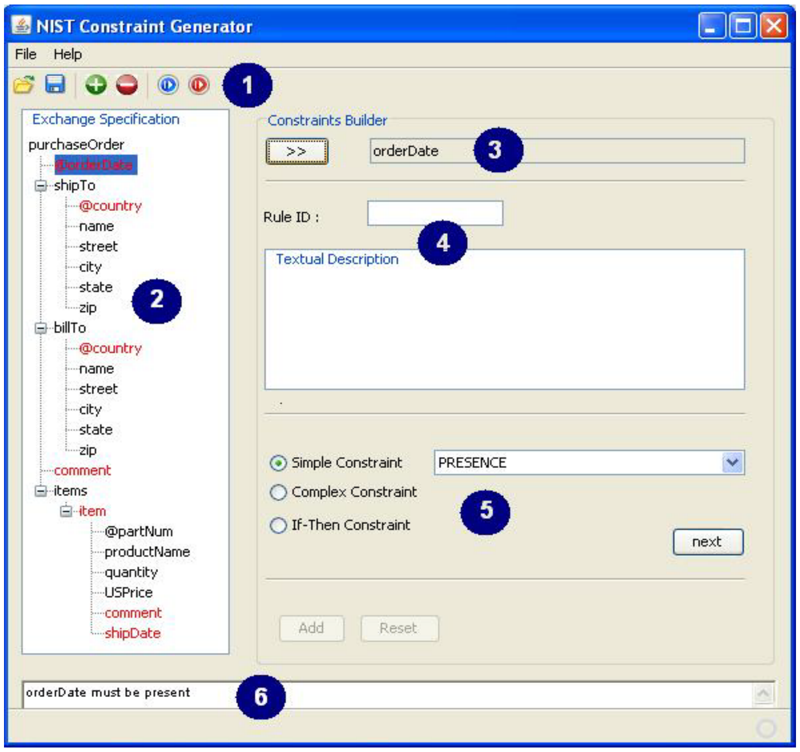
The user interface consists of two parts:

- The main window
- Constraint editor windows

The main window which helps the user track the constraints defined on the schema.  The constraint editor windows guide the user through the details of creating different types of constraints.  Each of these is described in detail below.

## 3.1) Main window

The main window of the Constraint Generator is divided into six areas.



orderDate

[✓ Simple Constraint] PRESENCE

orderDate must be present

Area 1: This is the tool bar. It contains buttons for opening/saving a project, expanding/collapsing a node, generating the   script, and generating the report.

Area 2: This is the panel showing the tree view of the schema. Note that color coding is used to highlight the nodes.

**Black** is the default color for a node.
**Red** indicates that a node is designated as an optional element in the schema.
**Green** indicates that a node has one or more constraints associated with it.
Gray indicates that a node is the context of a "must not be present" constraint.

Area 3 : This area is for choosing the context of the constraint, i.e. the node on which the constraint will be added.

Area 4 : This area provides text fields for specifying the ID and the textual description of the constraint to be added.

Area 5 : This is the panel for choosing the constraint to be added to a node. Once the desired constraint is chosen and the next button is hit, a dialog box will open showing the UI of the selected constraint. These interfaces are described below.

Area 6 : This area show messages resulting from the different actions performed by the user.

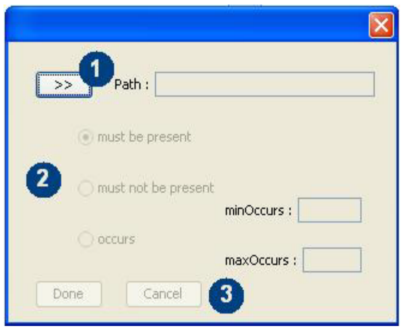
## 3.2) Constraint Interfaces

In this section we describe in detail the interfaces corresponding to the different types of constraints supported by the tool. Six types of constraints are supported:

- Presence
- Path in List
- Path in Range
- Path op Expression
- Complex Constraint
- IF-THEN Constraint

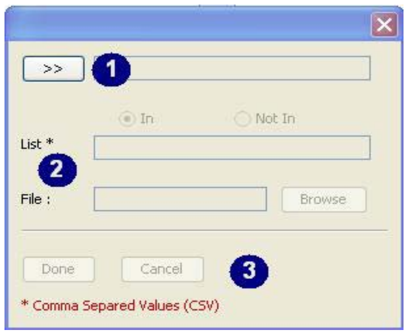
### 3.2.1) Presence

1. Button for selecting the node to be constrained
2. Presence constraint sub-type selector:
   - "must be present" requires that the node appear at least once
   - "must not be present" requires that the node not appear
   - "occurs" allows minimum and maximum occurrences of a node to be set.
3. Controllers for finishing or canceling the operation.



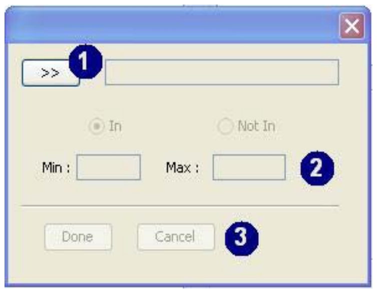
### 3.2.2) Path in List

1. Button for selecting the node to be constrained
2. List input fields :
   - Comma Separated Values (CSV)
   - Or a File containing CSV
3. Controllers for finishing or canceling the operation.



### 3.2.3) Path in Range

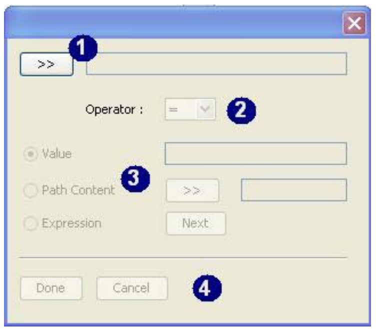
1. Button for selecting the node to be constrained
2. Minimum and Maximum Range input fields
3. Controllers for finishing or canceling the operation.



### 3.2.4) Path op Expression

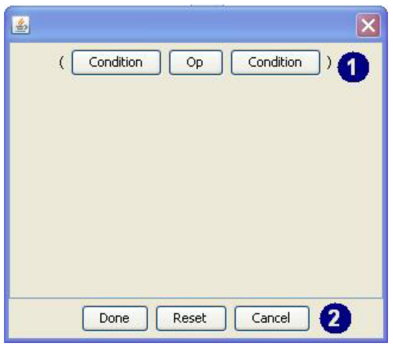
1. Button for selecting the node to be constrained
2. Operator selector :
   – equal "="
   – not equal "!="
3. Expression selector :
   – a specific value
   – the content of a path
   – an expression (combination of other paths content)

4.Controllers for finishing or canceling the operation.



### 3.2.5) Complex Constraint

1. Complex condition builder :
   - Condition button : Open a simple or a complex constraint dialog box
   - Op button : boolean operator
     - AND
     - OR
2. Controllers for finishing or canceling the operation.



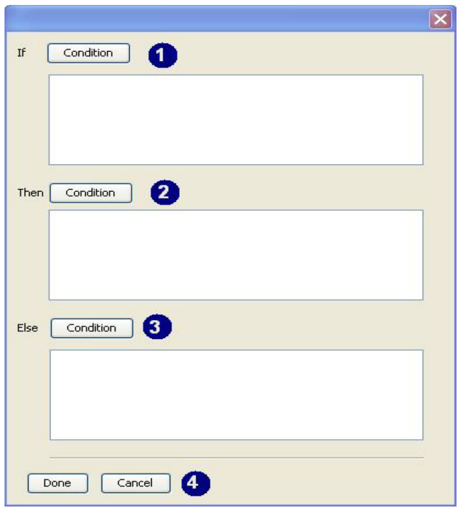
### 3.2.6) IF-THEN Constraint

1. Button for setting the condition. Once clicked, a dialog box for choosing the condition (simple or complex) will appear. The text area below it will show an overview of the condition.

2. Button for setting the condition to be verified, when the IF condition is true. Once clicked, a dialog box for choosing the condition (simple or complex) will appear. The text area below it will show an overview of this condition.

3. Button for setting the condition to be verified, when the IF condition is false. Once clicked, a dialog box for choosing the condition (simple or complex) will appear. The text area below it will show an overview of this condition.

4. Controllers for finishing or canceling the operation.



## 4) Basic Operations

This section describes basic operations of the Constraint Generator tool. Those operations include:

- opening a schema,
- saving/loading a project,
- expanding/collapsing a node,
- generating the Schematron script,
- generating the report and
- adding/removing a constraint.

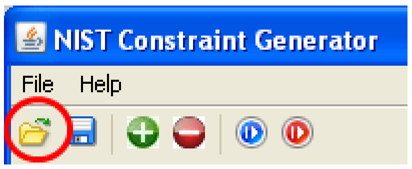
### 4.1) Opening a schema

**To open a schema :**
File → Open
OR
CTRL + O

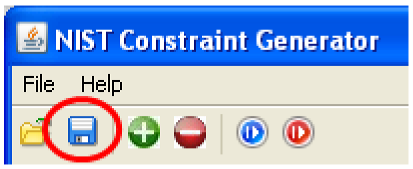
Click browse to navigate to the desired schema. Provide the name of the root element in the "Root Node Name" text field and click open. A tree view, illustrating the format of a potential instance file, will appear.

### 4.2) Loading/Saving a project

**To load a project :**
File → Open Project
OR
CTRL+SHIFT+O
OR
Click the "Open Project" icon

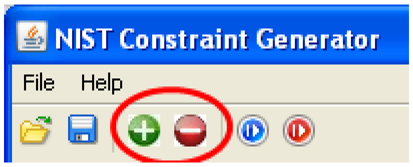


**To save a project:**
File → Save Project
OR
CTRL+S
OR
Click the "Save Project" icon



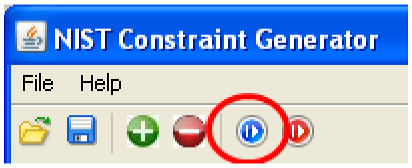
## 4.3) Expanding/Collapsing a node

**To expand/collapse a node:**
Click the +/- next to the node's name to expand or collapse it.
OR
Right click on the node, and select collapse or expand node on the drop down list
OR
Select the node, and then click the green "+" button on the tool bar to expand the node, or the red "-" to collapse it.

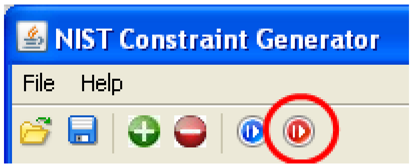


## 4.4) Generating the schematron script and the report

**To generate the schematron script:**
Click the "Generate Script" icon



**To generate the report:**
Click the "Generate Report" icon

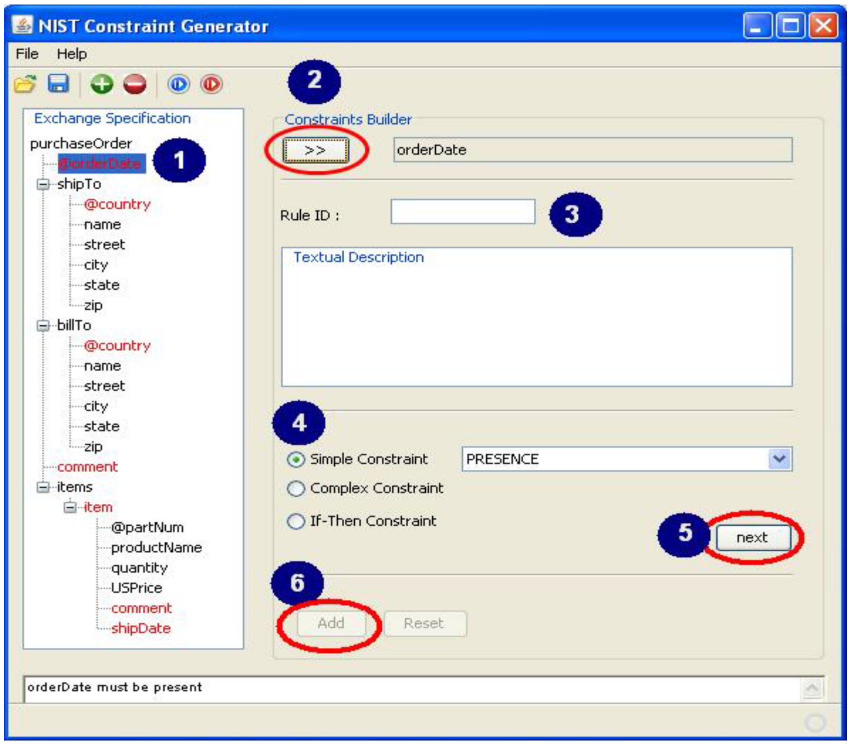


## 4.5) Adding a constraint

Adding constraint to a node is a multi-steps process.

Step 1 : Select the node to be constrained
Step 2 : Set the node as the context
Step 3 : Provide the ID and the textual description of the constraint
Step 4 : Select the type of the constraint to be added
Step 5 : Click the next button to open the dialog box of the selected constraint
Step 6 : Click on the Add button to add the constraint to the node or the Reset button to cancel the operation.

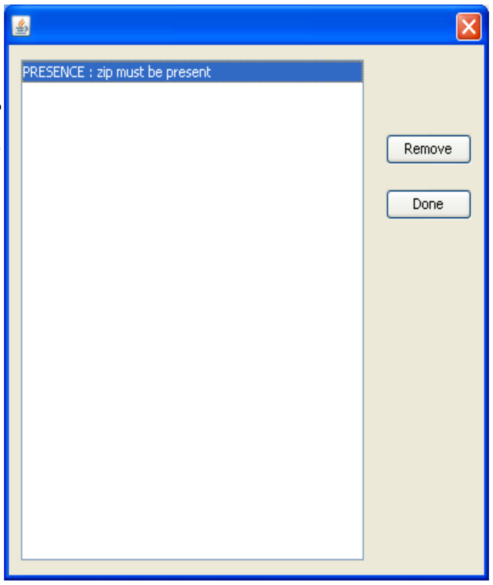
The picture below shows the different steps of the process.



orderDate must be present

## 4.6) Removing a constraint

To remove a constraint, select the node whose constraint should be deleted and right click. On the drop down menu, click “Delete Constraint.” A window will open, displaying all current constraints on the selected node. Highlight the constraint you want to remove, and click “Remove.”  Click “Done” when the constraint is removed from that element.



PRESENCE : zip must be present

Note : Multiple constraints selection is allowed by holding the CTRL command when selecting.

## 5) Example

This section shows an example for using the tool. We used the Purchase Order schema from W3C XML Schema Primer [6] which can be found at http://www.w3.org/TR/xmlschema-0/. We assumed that we want to constraint this schema for a specific need, defined the rules specification (the constraints that will be applied to this schema), and generated the Schematron scripts as well as a human readable document (the report) describing the rules. The following sections presented each of these items:

1. The Purchase Order XML Schema document
2. The rules specification
3. The generated script
4. The generated report

## 5.1) The Purchase Order XML Schema document

```xml
<?xml version="1.0"?>
 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:annotation>
   <xsd:documentation xml:lang="en">
   Purchase order schema for Example.com.
   Copyright 2000 Example.com. All rights reserved.
   </xsd:documentation>
 </xsd:annotation>
 <xsd:element name="purchaseOrder" type="PurchaseOrderType"/>
 <xsd:element name="comment" type="xsd:string"/>
 <xsd:complexType name="PurchaseOrderType">
   <xsd:sequence>
     <xsd:element name="shipTo" type="USAddress"/>
     <xsd:element name="billTo" type="USAddress"/>
     <xsd:element ref="comment" minOccurs="0"/>
     <xsd:element name="items" type="Items"/>
   </xsd:sequence>
   <xsd:attribute name="orderDate" type="xsd:date"/>
 </xsd:complexType>
 <xsd:complexType name="USAddress">
   <xsd:sequence>
     <xsd:element name="name" type="xsd:string"/>
     <xsd:element name="street" type="xsd:string"/>
     <xsd:element name="city" type="xsd:string"/>
     <xsd:element name="state" type="xsd:string"/>
     <xsd:element name="zip" type="xsd:decimal"/>
   </xsd:sequence>
   <xsd:attribute name="country" type="xsd:NMTOKEN" fixed="US"/>
 </xsd:complexType>
 <xsd:complexType name="Items">
   <xsd:sequence>
     <xsd:element name="item" minOccurs="0" maxOccurs="unbounded">
       <xsd:complexType>
         <xsd:sequence>
           <xsd:element name="productName" type="xsd:string"/>
           <xsd:element name="quantity">
             <xsd:simpleType>
               <xsd:restriction base="xsd:positiveInteger">
                 <xsd:maxExclusive value="100"/>
               </xsd:restriction>
             </xsd:simpleType>
           </xsd:element>
           <xsd:element name="USPrice" type="xsd:decimal"/>
           <xsd:element ref="comment" minOccurs="0"/>
           <xsd:element name="shipDate" type="xsd:date"
minOccurs="0"/>
         </xsd:sequence>
           <xsd:attribute name="partNum" type="SKU"
use="required"/>
       </xsd:complexType>
     </xsd:element>
   </xsd:sequence>
 </xsd:complexType>
 <!-- Stock Keeping Unit, a code for identifying products -->
 <xsd:simpleType name="SKU">
   <xsd:restriction base="xsd:string">
     <xsd:pattern value="\d{3}-[A-Z]{2}"/>
   </xsd:restriction>
 </xsd:simpleType>
</xsd:schema>
```

## 5.2) The Rules Specification

Rule 1 : Comment must be present under Purchase Order. (Presence)

Rule 2 : Comment must not be present under item. (Presence)

Rule 3 : The state in the shipping address must be either MD,DC or VA. (Path in List)

Rule 4 : The state in the billing address shouldn't be PA or NY. (Path in List)

Rule 5 : The zip code in the shipping address should be in the following range [00000, 50000]. (Path in Range)

Rule 6 : The zip code in the billing address shouldn't be in the following range [50000,100000]. (Path in Range)

Rule 7 : The country in the shipping address must be US. (Path op Expression)

Rule 8 : The country in the billing address shouldn't be FR. (Path op Expression)

Rule 9 : The order date should follow this pattern YYYY-MM-DD \d{4}\-\d{2}\-\d{2}. (Path match RegExp)

Rule 10 : The state in the shipping address should be present and its values should be either MD, VA or DC. (Complex Constraint)

Rule 11 : If the country in the billing address is US then the zip code must be present. (If then Constraint)

Rule 12 : If the country in the billing address is US then the zip code or the state must be present. (If then Constraint)

Rule 13 : If the city is present in the billing address then the state must be present otherwise the zip must be present. (If then Constraint)

## 5.3) The Generated Schematron Sscript

```
<?xml version="1.0" encoding="UTF-8" ?>
<schema queryBinding="xslt2" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       xmlns:xsl="http://www.w3.org/  1999/XSL/Transform"
xmlns="http://purl.oclc.org/dsdl/schematron">
<title>Schematron Rules generated by the NIST Constraint Generator</title>

<!-- ================================================================== -->
<!-- G L O B A L V A R I A B L E S -->
<!-- ================================================================== -->

<!-- ================================================================== -->
<!-- N A M E S P A C E S -->
<!-- ================================================================== -->

<!-- ================================================================== -->
<!-- R U L E S -->
<!-- ================================================================== -->

<pattern id="PATH_MATCH_REGEXP-9905727">
   <rule context="/purchaseOrder">
     <assert test="matches(string(@orderDate),'\d{4}\-\d{2}\-\d{2}')">
      R9 : The order date should follow this pattern YYYY-MM-DD.
     </assert>
   </rule>
</pattern>
<pattern id="PRESENCE-22024230">
   <rule context="/purchaseOrder/items[1]/item[1]">
     <assert test="count(*[local-name(.)='comment'][namespace-uri(.)=''])
&lt;= 0">
      R2 : Comment must not be present under item.
     </assert>
   </rule>
</pattern>
<pattern id="PATH_OP_EXPRESSION-22297541">
   <rule context="/purchaseOrder/shipTo[1]">
     <assert test="string(@country) = string('US')">
      R7 : The country in the shipping address must be US.
     </assert>
   </rule>
</pattern>
<pattern id="PATH_IN_RANGE-18886477">
   <rule context="/purchaseOrder/shipTo[1]/zip[1]">
     <assert test="number(.) &gt;= number(0) and number(.) &lt;=
number(50000)">
      R5 : The zip code in the shipping address should be in the following
range [00000,50000].
     </assert>
   </rule>
</pattern>
<pattern id="PATH_IN_LIST-104184">
   <rule context="/purchaseOrder/shipTo[1]/state[1]">
     <assert test="exists(index-of(tokenize('MD,DC,VA',','), string(.) ))">
      R3 : The state in the shipping address must be either MD,DC or VA.
     </assert>
   </rule>
</pattern>
<pattern id="Complex_Constraint-6193963">
```

```
   <rule context="/purchaseOrder/shipTo[1]/state[1]">
     <assert test="(count(/purchaseOrder/shipTo[1]/state[1]) > 0) and
       (exists(index-of(tokenize('MD,DC,VA',','),
string(/purchaseOrder/shipTo[1]/state[1]) )))">
      R10 : The state in the shipping address should be present and its values
should be
      either MD, VA or DC.
     </assert>
   </rule>
</pattern>
<pattern id="PRESENCE-16104144">
   <rule context="/purchaseOrder">
     <assert test="count(*[local-name(.)='comment'][namespace-uri(.)='']) >
0">
      R1 : Comment must be present under Purchase Order.
     </assert>
   </rule>
</pattern>
<pattern id="PATH_OP_EXPRESSION-12310589">
   <rule context="/purchaseOrder/billTo[1]">
     <assert test="string(@country) != string('FR')">
      R8 : The country in the billing address shouldn't be FR.
     </assert>
   </rule>
</pattern>
<pattern id="IF-Then_Constraint-7441969">
   <rule context="/purchaseOrder/billTo[1]">
     <report test="((string(/purchaseOrder/billTo[1]/@country) = string('US'))
and
        not(count(/purchaseOrder/billTo[1]/zip[1]) > 0))">
      R11 : If the country in the billing address is US then the zip code must
be present.
     </report>
   </rule>
</pattern>
<pattern id="IF-Then_Constraint-19820996">
   <rule context="/purchaseOrder/billTo[1]">
     <report test="((string(/purchaseOrder/billTo[1]/@country) = string('US'))
and
     not((count(/purchaseOrder/billTo[1]/zip[1]) > 0) or
(count(/purchaseOrder/billTo[1]/state[1])
     > 0)))">
      R12 : If the country in the billing address is US then the zip code or
the state must be present.
     </report>
   </rule>
</pattern>
<pattern id="PATH_IN_RANGE-3085858">
   <rule context="/purchaseOrder/billTo[1]/zip[1]">
     <assert test="number(.) &lt; number(50000) or number(.) &gt;
number(10000)">
      R6 : The zip code in the billing address shouldn't be in the following
range [50000,100000].
     </assert>
</rule>
</pattern>
<pattern id="PATH_IN_LIST-22434271">
   <rule context="/purchaseOrder/billTo[1]/state[1]">
```

```
     <assert test="not(exists(index-of(tokenize('PA,NY',','), string(.) )))">
      R4 : The state in the billing address shouldn't be PA or NY.
     </assert>
   </rule>
</pattern>
<pattern id="IF-Then_Constraint-1655982">
   <rule context="/purchaseOrder/billTo[1]/city[1]">
     <report test="(((count(/purchaseOrder/billTo[1]/city[1]) > 0) and
      not(count(/purchaseOrder/billTo[1]/state[1]) > 0)) or
      (not(count(/purchaseOrder/billTo[1]/city[1]) > 0) and
      not(count(/purchaseOrder/billTo[1]/zip[1]) > 0)))">
      R13 : If the city is present in the billing address then the state must
be present otherwise the
      zip must be present.
     </report>
   </rule>
</pattern>
</schema>
```

## 5.4) The Generated Report

| Rule ID | Description | Rule Category | Condition | Rule Type | Comments |
|---|---|---|---|---|---|
| | **Context : /purchaseOrder/@orderDate** | | | | |
| R9 | The order date should follow this pattern YYYY-MM-DD. | Path match RegExp | matches(string(/purchase | assert | |
| | **Context : /purchaseOrder/shipTo[1]/@country** | | | | |
| R7 | The country in the shipping address must be US. | Path op Expression | string(/purchaseOrder/shi | assert | |
| | **Context : /purchaseOrder/shipTo[1]/state[1]** | | | | |
| R3 | The state in the shipping address must be either MD,DC or VA. | Path in List | exists(index-of(tokenize(' | assert | |
| R10 | The state in the shipping address should be present and its values should be either MD, VA or DC. | Complex_Constraint | (count(/purchaseOrder/sh | assert | |
| | **Context : /purchaseOrder/shipTo[1]/zip[1]** | | | | |
| R5 | The zip code in the shipping address should be in the following range [00000, 50000]. | Path in Range | number(/purchaseOrder/s | assert | |
| | **Context : /purchaseOrder/billTo[1]/@country** | | | | |
| R8 | The country in the billing address shouldn't be FR. | Path op Expression | string(/purchaseOrder/bill | assert | |
| R11 | If the country in the billing address is US then the zip code must be present. | IF-Then_Constraint | string(/purchaseOrder/bill | report | |
| R12 | If the country in the billing address is US then the zip code or the state must be present. | IF-Then_Constraint | string(/purchaseOrder/bill | report | |
| | **Context : /purchaseOrder/billTo[1]/city[1]** | | | | |
| R13 | If the city is present in the billing address then the state must be present otherwise the zip must be present. | IF-Then_Constraint | count(/purchaseOrder/bill | report | |
| | **Context : /purchaseOrder/billTo[1]/state[1]** | | | | |
| R4 | The state in the billing address shouldn't be PA or NY. | Path in List | not(exists(index-of(tokeni | assert | |
| | **Context : /purchaseOrder/billTo[1]/zip[1]** | | | | |
| R6 | The zip code in the billing address shouldn't be in the following range [50000,100000]. | Path in Range | number(/purchaseOrder/b | assert | |
| | **Context : /purchaseOrder/comment[1]** | | | | |
| R1 | Comment must be present under Purchase Order. | Presence | count(/purchaseOrder/co | assert | |
| | **Context : /purchaseOrder/items[1]/item[1]/comment[1]** | | | | |
| R2 | Comment must not be present under item. | Presence | count(/purchaseOrder/ite | assert | |

# Disclaimer

## References

1. R. Jelliffe; *Schematron*;  last updated August 7, 2008; [cited 2008]; available from http://www.schematron.com/.
2. World Wide Web Consortium; *Extensible Markup Language (XML)*; [cited 2008]; available from http://www.w3.org/XML/.
3. World Wide Web Constortium; *XML Schema 1.0*; [cited 2008]; available from http://www.w3.org/XML/Schema.
4. *Java*; [cited 2008]; available from http://java.sun.com/.
5. *MSID XML Testbed Home Page*; [cited September 3, 2008]; available from http://www.mel.nist.gov/msid/XML_testbed/index.html.
6. World Wide Web Constortium; *XML Schema Part 0:  Primer Second Edition*; World Wide Web Constortium; October 28, 2004; available from http://www.w3.org/TR/xmlschema-0/.