Improvements in WSOL Grammar and "Premier" WSOL Parser

Kruti Patel, Bernard Pagurek, Vladimir Tosic

Research Report SCE-03-25
October 2003

The Department of Systems and
Computer Engineering,
Carleton University, Ottawa, Canada
October 2003

Abstract

This research report presents the recent research work that has been done on the WSOL (Web Service Offerings Language) grammar and the "Premier" WSOL parser.

The WSOL grammar has been extended in several ways. First, specification of periodic QoS constraints was introduced and the specification of QoS constraints was improved by adding evaluation period for occasional QoS constraint evaluation. Next, dynamic relationships between service offerings can now include complex expressions. Further, naming of all kinds of WSOL expressions and referencing all kinds of WSOL expressions from other WSOL constraints was added. The specification of subscription information, validity period information, and auto manipulation information for WSOL service offerings were added. The WSOL grammar has also been improved in order to enable specification of existing types of WSOL statements and additional penalty statements using the general <*statement*> construct and specification of CGs and CGTs in which only some constraints have to be satisfied. We use the version number 1.1 to denote this new version of WSOL.

The existing WSOL example has been improved and some new WSOL examples have been developed in this research work using the improved and extended WSOL grammar. A WSOL example for the specification of future-conditions (the XML grammar for which was developed prior to this research work) has also been created.

The "Premier" WSOL parser has been extended and improved to validate specifications of future-conditions and service offerings dynamic relationships. Further, it was improved to validate some of the new WSOL specifications developed using the improved WSOL grammar and to detect and report syntax and semantic errors in these specifications. For example, the "Premier" WSOL parser was extended for the specification of periodic QoS constraints, the extended specification of QoS constraints, dynamic relationships between service offerings that include complex expressions, and WSOL statements and additional penalty statements specified using the general <*statement*> construct. All these improvements and extensions in the WSOL grammar and the WSOL parser have been verified using the improved WSOL example and the newly developed WSOL examples.

Table of Contents

List of Figures

1 Introduction

The Web Service Offerings Language (WSOL) is a new XML-based language for the formal specification of various types of constraint, management statements, and multiple classes of service for Web Services. It is compatible with the Web Service Description Language (WSDL) version 1.1. It can be used both for Web Service management and Web Service composition management. In Kruti Patel's Master's thesis "XML Grammar and Parser for the Web Service Offerings Language" [1], she had developed the XML grammar for WSOL and a parser named "Premier" for WSOL in Java. The WSOL grammar enables generation of WSOL documents that conform to the grammar and the WSOL parser enables validation of WSOL and WSOL-related documents and detection and notification of various syntax and semantic errors in them. An illustrative WSOL example was also developed that enables verification of the developed WSOL grammar and the WSOL parser.

The research work presented in this research report contains extensions and improvements of the WSOL grammar and the "Premier" WSOL parser. Since the WSOL language has been improved conceptually, the corresponding improvements in the WSOL grammar and the "Premier" WSOL parser were essential. Also, several WSOL concepts were already defined during Kruti's Master's thesis work but not implemented in the WSOL parser due to time limits. Therefore, it was necessary to improve the WSOL parser. The WSOL example has also been improved to enable verification of the improved WSOL grammar and the WSOL parser. Throughout this document, the old version of WSOL language is referred to as WSOL 1.0 and the new improved version is referred to as WSOL 1.1.

Justification for some of these improvements was presented in our recent publications about WSOL [2, 3, 4, 5]. The management applications of the WSOL language (and, thus, some of these improvements) were described in [5, 6, 7].

In this research report, the improvements done in the category of WSOL constraints are presented first, followed by an explanation of the improvements in Service Offerings Dynamic Relationships (SODR), WSOL expressions, WSOL statements, WSOL reusability constructs, and WSOL service offerings. The appendices contain the complete XML grammar for WSOL 1.1 and related files (e.g., ontologies), including the parts that have not changed from WSOL 1.0.

All the schema diagrams (i.e., graphical representations of various schemas) given in this report are generated with XMLSpy Schema Editor (XMLSpy is an XML development tool from Altova (http://www.xmlspy.com)). An explanation of some symbols used by the XMLSpy Schema Editor to generate schema diagrams is given in **Appendix G**.

2 WSOL Constraints

WSOL 1.0 enabled specification of functional constraints (pre-conditions, post-conditions, and future-conditions), non-functional constraints (i.e., QoS constraints), and access rights. The new version of the WSOL language, i.e., WSOL 1.1, also enables specification of periodic QoS constraints. For future-conditions, that were part of WSOL 1.0, an example was defined and the "Premier" WSOL parser was enabled to check them. In addition, the specification of QoS constraints is extended to enable occasional evaluation of QoS constraints.

2.1 Periodic QoS Constraints

A periodic QoS constraint is a QoS constraint that is evaluated periodically between a given start and stop date and no more than the given maximum number of repetitions. The XML grammar that enables specification of a WSOL periodic QoS constraint is defined in a schema called *"PeriodicQoSSchema.xsd"* (Figure 1 and Figure 2).

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/PeriodicQoSSchema"
xmlns:periodicQoSSchema="http://www.sce.carleton.ca/~kpatel/WSOL/PeriodicQoSSchema"
xmlns:wsol="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchem
a" xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/PeriodicQoSSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xsd:import
  namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
  schemaLocation="Schemas/ExpressionSchema.xsd"/>
  <xsd:import
  namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
  schemaLocation="Schemas/WSOLSchema.xsd"/>
  <xsd:complexType name="periodicQoS">
    <xsd:complexContent>
    <xsd:extension base="wsol:constraintType">
      <xsd:sequence>
        <xsd:element ref="evaluationStartTime"/>
        <xsd:element ref="evaluationPeriod"/>
        <xsd:element ref="evaluationStopTime"/>
        <xsd:element ref="maxRepetitions"/>
        <xsd:element ref="expressionSchema:booleanExpression"/>
      </xsd:sequence>
    </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="evaluationStartTime">
    <xsd:complexType>
      <xsd:attribute name="time" type="xsd:dateTime" use="required"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="evaluationPeriod">
    <xsd:complexType>
      <xsd:attribute name="interval" type="xsd:duration" use="required"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="evaluationStopTime">
    <xsd:complexType>
      <xsd:attribute name="time" type="xsd:dateTime" use="required"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="maxRepetitions">
    <xsd:complexType>
      <xsd:attribute name="number" type="xsd:int" use="required"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Figure 1 Periodic QoS Schema (Grammar for the specification of a WSOL periodic QoS constraint)

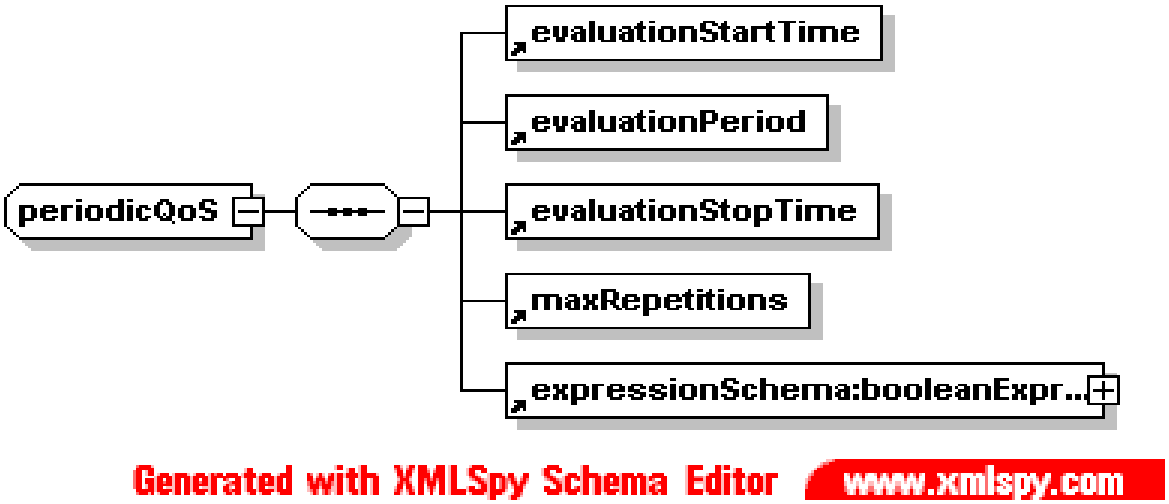


Figure 2 Periodic QoS Schema (Graphical representation of the grammar for the specification of a WSOL periodic QoS constraint)

The grammar for the specification of a WSOL periodic QoS constraint enables specification of the start time (i.e., the time when the periodic evaluation of the specified QoS constraint will begin), the period (i.e., the time duration after which the specified QoS constraint will be evaluated), the stop time (i.e., the time when the periodic evaluation of the specified QoS constraint will stop), maximum repetitions (i.e., the maximum number of times when the specified QoS constraint will be evaluated), and a Boolean expression (i.e., the QoS constraint to be evaluated). Figure 3 shows an example of a WSOL periodic QoS constraint.

```
<wsol:constraint name="perQoScons1"
xsi:type="periodicQoSSchema:periodicQoS" service="WSOL-ANY">
  <periodicQoSSchema:evaluationStartTime time="2003-11-01T00:00:00-05:00"/>
  <periodicQoSSchema:evaluationPeriod interval="P1D"/>
  <periodicQoSSchema:evaluationStopTime time="2004-11-01T00:00:00-05:00"/>
  <periodicQoSSchema:maxRepetitions number="366"/>
  <expressionSchema:booleanExpression>
    <expressionSchema:arithmeticWithUnitExpression>
      <expressionSchema:QoSmetric metricType="QoSMetricOntology:DailyAvailability"
      metricUnit="QoSMeasOntology:Percentage" service="WSOL-ANY"
      portOrPortType="WSOL-ANY" operation="WSOL-ANY"
      measuredBy="WSOL_INTERNAL"/>
    </expressionSchema:arithmeticWithUnitExpression>
    <expressionSchema:arithmeticWithUnitComparator type=">"/>
    <expressionSchema:arithmeticWithUnitExpression>
      <wsol:numberWithUnitConstant>
        <wsol:number value="95"/>
        <wsol:unit type="QoSMeasOntology:Percentage"/>
      </wsol:numberWithUnitConstant>
    </expressionSchema:arithmeticWithUnitExpression>
  </expressionSchema:booleanExpression>
</wsol:constraint>
```

Figure 3 Example of a WSOL periodic QoS Constraint

In the above example, a periodic QoS constraint named "*perQoScons1*" is specified. The applicability domain for this periodic QoS constraint, that is specified with the domain attribute "*service*", is "WSOL-ANY" and it means that the specified periodic QoS constraint could be applicable to any Web Service. (Only the domain attribute "*service*" is used to specify the applicability domain for a periodic QoS constraint. The domain attributes *"portOrPortType"* and *"operation"* are not used to specify the applicability domain for a periodic QoS constraint because a periodic QoS constraint is not related to invocation of a particular operation.) The example periodic QoS constraint "*perQoScons1*" states that the measured value of the QoS metric "*DailyAvailability*" (defined in an external ontology of QoS metrics) must be greater than 95 Percentage. This periodic QoS constraint is evaluated daily, at midnight, between the given start and stop date and no more than the given maximum number of repetitions.

The "Premier" WSOL parser has been improved to validate WSOL periodic QoS constraints. While validating, the WSOL parser stores important data (such as, the specified periodic QoS constraint's applicability domain, evaluation start and stop time, evaluation period, maximum number of repetitions, and the Boolean expression) into a symbol table. The WSOL parser also detects and reports syntax as well as semantic errors in the WSOL periodic QoS constraints. A few examples of such errors are given below:

- the type of constraint specified as the value of "*xsi:type*" attribute is incorrect,
- the applicability domain attribute "*service*" is not specified,
- the periodic QoS constraint evaluation start or stop time is not specified,
- the periodic QoS constraint evaluation period is not specified,

- the maximum number of repetitions for the periodic QoS constraint evaluation is not specified,
- the type of QoS metric specified is incorrect, and
- the unit of QoS metric specified is incorrect.

2.2 Future Conditions

A future-condition is similar to a post-condition except that it is evaluated on a specific date/time or after a specific duration of time after the execution of the invoked operation is over. We have introduced the concept of future-conditions in [2]. The XML grammar that enables specification of a WSOL future-condition was already defined in WSOL 1.0 in a schema called *"FutureConditionSchema.xsd"* (Figure 4 and Figure 5). But, no example of specification of future-condition had been developed earlier. Also, the implementation of validation of the specifications of future-conditions was not built into the "Premier" WSOL parser. In this research work, an example of specification of future-condition has been developed and the "Premier" WSOL parser has been improved to validate the specifications of future-conditions.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/FutureConditionSchema"
xmlns:futureConditionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/FutureConditionSche
ma" xmlns:wsol="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchem
a" xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/FutureConditionSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xsd:import
  namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
  schemaLocation="Schemas/ExpressionSchema.xsd"/>
  <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
  schemaLocation="Schemas/WSOLSchema.xsd"/>
  <xsd:complexType name="futureCondition">
    <xsd:complexContent>
      <xsd:extension base="wsol:constraintType">
        <xsd:sequence>
          <xsd:choice>
            <xsd:element ref="evaluateAt"/>
            <xsd:element ref="evaluateAfter"/>
          </xsd:choice>
          <xsd:element ref="expressionSchema:booleanExpression"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="evaluateAt">
    <xsd:complexType>
      <xsd:attribute name="dateTime" type="xsd:dateTime" use="required"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="evaluateAfter">
    <xsd:complexType>
      <xsd:attribute name="duration" type="xsd:duration" use="required"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Figure 4 Future Condition Schema (Grammar for the specification of a WSOL future-condition)

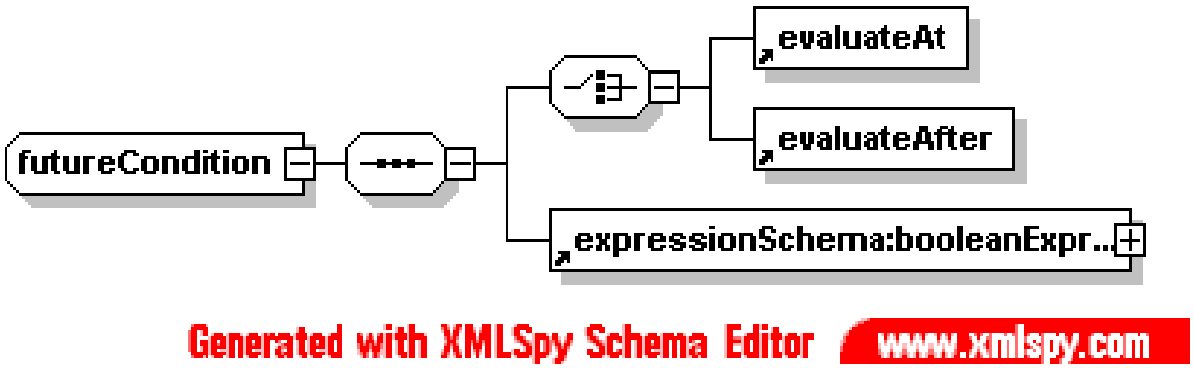


Figure 5 Future Condition Schema (Graphical representation of the grammar for the specification of a WSOL future-condition)

A specification of a WSOL future-condition contains a specific date/time (after the execution of the invoked operation is over) when the specified future-condition will be evaluated or a specific duration of time that has to elapse from the end of invocation to the evaluation of the post-condition. In addition, a future-condition contains a Boolean expression that represents the future-condition to be evaluated. Figure 6 shows an example of a WSOL future-condition.

```
<wsol:serviceOffering name="SO1" service="buyStock:buyStockService">
  <wsol:constraint name="futureCondition1" xsi:type="futureConditionSchema:futureCondition"
  service="buyStock:buyStockService" portOrPortType="buyStock:buyStockServicePort"
  operation="buyStock:buySingleStockOperation">
    <futureConditionSchema:evaluateAfter duration="P2D"/>
    <expressionSchema:booleanExpression>
      <expressionSchema:booleanExpression>
        <expressionSchema:booleanVariable
        bvName="buyStock:buySingleStockResponse.stockCertificatesMailed"/>
      </expressionSchema:booleanExpression>
      <expressionSchema:booleanComparator type="=="/>
      <expressionSchema:booleanExpression>
        <expressionSchema:booleanConstant type="true"/>
      </expressionSchema:booleanExpression>
    </expressionSchema:booleanExpression>
  </wsol:constraint>
</wsol:serviceOffering>
```

Figure 6 Example of a WSOL future-condition

In the above example, a future-condition named *"futureCondition1"* is specified for a particular service (buyStockService), port (buyStockServicePort), and operation (buySingleStockOperation). This example future-condition states that the value of the Boolean variable *"buyStock:buySingleStockResponse.stockCertificatesMailed"* should be true when it is evaluated (i.e., after 2 days after the execution of the invoked operation buySingleStockOperation is over).

The "Premier" WSOL parser has been improved to validate the WSOL future-conditions. While validating, the WSOL parser stores important data (such as, the specified future-condition's applicability domain, evaluation date or time or evaluation duration, and the Boolean expression) into a symbol table. The WSOL parser also detects and reports syntax as well as semantic errors in the WSOL future-conditions. A few examples of such errors are given below:

- The type of constraint specified as the value of "*xsi:type*" attribute is incorrect,
- The applicability domain attribute "*service*" is not specified or is incorrect,
- The applicability domain attribute "*portOrPortType*" is not specified or is incorrect,
- The applicability domain attribute "*operation*" is not specified or is incorrect,
- The future-condition evaluation date/time is not specified,
- The future-condition evaluation duration is not specified,
- The variable (i.e., Boolean variable, or arithmetic variable, or string variable, or any other type of variable) on which the future-condition is to be evaluated is not specified or is incorrect, and
- The constant (i.e., Boolean constant, or arithmetic constant, or string constant, or any other type of constant) on which the future-condition is to be evaluated is not specified or is incorrect.

2.3 Occasional Evaluation of QoS constraints

WSOL 1.1 also enables specification of the evaluation period for QoS constraints. Such information is essential for occasional evaluation of QoS constraints. The QoS constraints for which the evaluation period is specified are not evaluated always but for some random operation invocations determined by the accounting party. This reduces the overhead of run-time monitoring of QoS constraints, but it also means that when such QoS constraint is not checked its violation will not be detected. To prevent misuse, it is advisable that such QoS constraints are evaluated by a third party is evaluating and the service provider does not know whether for a particular invocation the constraint will be evaluated or not. The concept of occassional evaluation of constraints and its benefits were also discussed in [3].

In order to enable specification of the evaluation period for QoS constraints, the XML grammar that enables specification of a WSOL QoS constraint has been improved. An additional attribute named *"evalPeriod"* has been added to the WSOL QoS constraint grammar, as shown in Figure 7 in bold. The attribute *"evalPeriod"* is of type *"positiveInteger"* i.e., its value can only be 1 or more. The attribute *"evalPeriod"* is optional and its default value is "1". So, if the attribute *"evalPeriod"* is not specified in a QoS constraint then it is assumed that the evaluation period is "1" and that the constraint is to be evaluated for every operation invocation.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/QoS_Schema"
xmlns:qosSchema="http://www.sce.carleton.ca/~kpatel/WSOL/QoS_Schema"
xmlns:wsol="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/QoS_Schema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
   schemaLocation="Schemas/ExpressionSchema.xsd"/>
   <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
   schemaLocation="Schemas/WSOLSchema.xsd"/>
   <xsd:complexType name="QoS">
      <xsd:complexContent>
         <xsd:extension base="wsol:constraintType">
            <xsd:sequence>
               <xsd:element ref="expressionSchema:booleanExpression"/>
            </xsd:sequence>
            <xsd:attribute name="evalPeriod" type="xsd:positiveInteger" use="optional"
            default="1"/>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
</xsd:schema>
```

Figure 7 QoS Schema (Shows addition of attribute “evalPeriod” to the WSOL QoS constraint grammar)

An example of a WSOL QoS constraint for which the evaluation period is specified is shown in Figure 8.

```
<wsol:constraint name="QoScons1" xsi:type="qosSchema:QoS"
service="buyStock:buyStockService" portOrPortType="buyStock:buyStockServicePort"
operation="buyStock:buySingleStockOperation" evalPeriod="5">
  <expressionSchema:booleanExpression name="BE1">
    <expressionSchema:arithmeticWithUnitExpression>
      <expressionSchema:QoSmetric metricType="QoSMetricOntology:ResponseTime"
      metricUnit="QoSMeasOntology:millisecond" service="buyStock:buyStockService"
      portOrPortType="buyStock:buyStockServicePort"
      operation="buyStock:buySingleStockOperation" measuredBy="WSOL_INTERNAL"/>
    </expressionSchema:arithmeticWithUnitExpression>
    <expressionSchema:arithmeticWithUnitComparator type="&lt;"/>
    <expressionSchema:arithmeticWithUnitExpression>
      <wsol:numberWithUnitConstant>
        <wsol:number value="10"/>
        <wsol:unit type="QoSMeasOntology:millisecond"/>
      </wsol:numberWithUnitConstant>
    </expressionSchema:arithmeticWithUnitExpression>
  </expressionSchema:booleanExpression>
</wsol:constraint>
```

Figure 8 Example of a WSOL QoS Constraint for which the evaluation period is specified using attribute "evalPeriod"

In this WSOL QoS constraint example, the attribute *"evalPeriod"* is specified (shown in bold) with a value of "5" which means that this QoS constraint is evaluated only for some operation invocations determined randomly and on an average for 1/5 operation invocations. The "Premier" WSOL parser has also been extended to validate the WSOL QoS constraints for which the evaluation period is specified.

3 Service Offerings Dynamic Relationships (SODR)

3.1 Validation of specifications of dynamic relationships between service offerings

The WSOL grammar that enables specification of dynamic relationships between service offerings was already developed earlier as part of Kruti Patel's Master's thesis [1]. An example of specification of dynamic relationships between service offerings was also developed earlier and is shown here in Figure 9.

```
<?xml version="1.0" encoding="UTF-8"?>
<sodr:SerOffDynRels xmlns:sodr="http://www.sce.carleton.ca/~kpatel/WSOL/SODRSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.sce.carleton.ca/~kpatel/WSOL/SODRSchema SODRSchema.xsd"
xmlns:serviceOfferings="http://www.sce.carleton.ca/~kpatel/WSOL/serviceOfferings"
xmlns:buyStock="http://www.sce.carleton.ca/~kpatel/buyStock/buyStockService"
service="buyStock:buyStockService">
    <sodr:SerOffDynRel sodrName="SODR1">
        <sodr:currentSO name="serviceOfferings:SO1"/>
        <sodr:unsatisfiedConstraints>
            <sodr:unsatisfiedConstraintRef name="serviceOfferings:SO1.C3"/>
            <sodr:unsatisfiedConstraintRef name="serviceOfferings:SO1.C5"/>
            <!--<sodr:unsatisfiedConstraintRef name = "serviceOfferings:SO1.CG6"/>-->
        </sodr:unsatisfiedConstraints>
        <sodr:replacementSO name="serviceOfferings:SO2"/>
    </sodr:SerOffDynRel>
</sodr:SerOffDynRels>
```

Figure 9 Example of specification of dynamic relationships between service offerings

These specifications of dynamic relationships between service offerings could not be validated earlier by the "Premier" WSOL parser since this feature was not yet implemented in the parser. However, the "Premier" WSOL parser has been now improved to validate the specifications of dynamic relationships between service offerings. While validating, the WSOL parser stores important data (such as, the name of the dynamic relationship between two service offerings, the name of the current service offering, the names of unsatisfied constraints from the current service offering, and the name of the replacement service offering) into a symbol table. The WSOL parser also detects and reports syntax as well as semantic errors in these specifications. A few examples of such errors are given below:

- the name of the service offerings dynamic relationship is not specified,
- the name of the current service offering is not specified,
- the name of the current service offering is specified but it does not exist,
- the names of unsatisfied constraints from the current service offering are not specified,
- the names of unsatisfied constraints from the current service offering are specified but are incorrect,
- the unsatisfied constraints specified do not belong to the current service offering,
- the name of the replacement service offering is not specified, and
- the name of the replacement service offering is specified but it does not exist.

In the specifications of dynamic relationships between two service offerings, the WSOL language supports referencing of constraints as well as constraint groups (CGs) in the set of unsatisfied constraints specified for a current service offering. Even though referencing of constraint groups (CGs) in SODRs is supported by WSOL, it is not recommended since constraint groups may be too complex to keep track of by a WSOL management infrastructure. Maybe referencing of constraint groups in SODRs will be disabled in future versions of WSOL.

3.2 Inclusion of complex expressions in the specification of dynamic relationships between service offerings

WSOL 1.1 enables inclusion of complex expressions in the specification of dynamic relationships between service offerings. All the unsatisfied constraints from the current service offering can now be specified within a single Boolean expression with complex relationships with each other, for example, *"(C1 OR C2) AND C3"*. Earlier, all the unsatisfied constraints from the current service offering were only specified in a sequence and so it was assumed that there was only an *"AND"* relationship between them. Now, using the improved XML grammar for the specification of dynamic relationships between service offerings (Figure 10 and Figure 11), all the unsatisfied constraints from the current service offering could either be specified in a sequence or in the form of a complex expression. The need for the improved specification of dynamic relationships between service offerings was also discussed in [5].

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/SODRSchema"
xmlns:sodr="http://www.sce.carleton.ca/~kpatel/WSOL/SODRSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/SODRSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xsd:import
    namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
    schemaLocation="Schemas/ExpressionSchema.xsd"/>
    <xsd:element name="SerOffDynRels">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="SerOffDynRel" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="service" type="xsd:QName" use="required"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="SerOffDynRel">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="currentSO"/>
                <xsd:element ref="unsatisfiedConstraints"/>
                <xsd:element ref="replacementSO"/>
            </xsd:sequence>
            <xsd:attribute name="sodrName" type="xsd:NCName" use="required"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="currentSO">
        <xsd:complexType>
            <xsd:attribute name="name" type="xsd:QName" use="required"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="replacementSO">
        <xsd:complexType>
            <xsd:attribute name="name" type="xsd:QName" use="required"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="unsatisfiedConstraints">
        <xsd:complexType>
            <xsd:choice>
                <xsd:element ref="unsatisfiedConstraintRef" maxOccurs="unbounded"/>
                <xsd:element ref="expressionSchema:booleanExpression"/>
            </xsd:choice>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="unsatisfiedConstraintRef">
        <xsd:complexType>
            <xsd:attribute name="name" type="xsd:QName" use="required"/>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Figure 10 SODR Schema (Improved XML grammar that supports inclusion of complex expressions in the specification of dynamic relationships between service offerings)

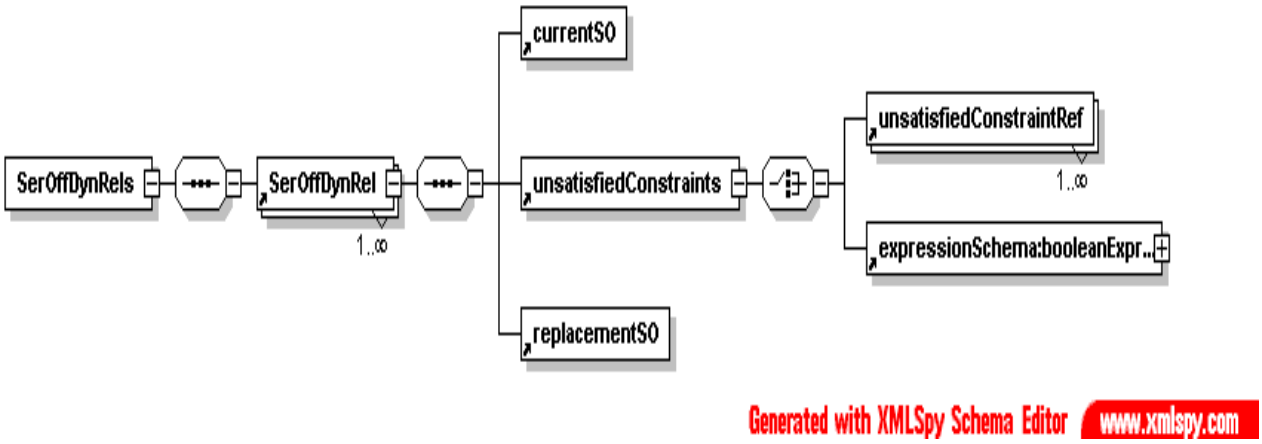


Figure 11 SODR Schema (Graphical representation)

Figure 12 shows an example of inclusion of complex expressions in the specification of dynamic relationships between service offerings.

```
<?xml version="1.0" encoding="UTF-8"?>
<sodr:SerOffDynRels xmlns:sodr="http://www.sce.carleton.ca/~kpatel/WSOL/SODRSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.sce.carleton.ca/~kpatel/WSOL/SODRSchema SODRSchema.xsd
http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema Schemas/ExpressionSchema.xsd"
xmlns:serviceOfferings="http://www.sce.carleton.ca/~kpatel/WSOL/serviceOfferings"
xmlns:buyStock="http://www.sce.carleton.ca/~kpatel/buyStock/buyStockService"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
service="buyStock:buyStockService">
    <sodr:SerOffDynRel sodrName="SODR1">
        <sodr:currentSO name="serviceOfferings:SO1"/>
        <sodr:unsatisfiedConstraints>
            <expressionSchema:booleanExpression>
                <expressionSchema:booleanExpression>
                    <sodr:unsatisfiedConstraintRef name="serviceOfferings:SO1.C3"/>
                </expressionSchema:booleanExpression>
                <expressionSchema:binaryBooleanOperator type="OR"/>
                <expressionSchema:booleanExpression>
                    <sodr:unsatisfiedConstraintRef name="serviceOfferings:SO1.C5"/>
                </expressionSchema:booleanExpression>
            </expressionSchema:booleanExpression>
        </sodr:unsatisfiedConstraints>
        <sodr:replacementSO name="serviceOfferings:SO2"/>
    </sodr:SerOffDynRel>
</sodr:SerOffDynRels>
```

Figure 12 Example of inclusion of complex expressions in the specification of Dynamic Relationships between Service Offerings

In the above example, a service offerings dynamic relationship named *"SODR1"* is specified between two service offerings *"SO1"* (the current service offering) and *"SO2"* (the replacement service offering). Two unsatisfied constraints from the current service offering *"SO1"* are specified within a single Boolean expression with complex relationship with each other, i.e., *"serviceOfferings:SO1.C3* ***OR*** *serviceOfferings:SO1.C5"* which means that if constraint *"serviceOfferings:SO1.C3"* is unsatisfied **or** if constraint *"serviceOfferings:SO1.C5"* is unsatisfied then the current service offering *"SO1"* should be replaced by the replacement service offering *"SO2"*.

The "Premier" WSOL parser has been improved to validate the specifications of dynamic relationships between service offerings that include complex expressions. While validating, the WSOL parser stores important data (such as, the included complex expression) into a symbol table. The WSOL parser also detects and reports syntax as well as semantic errors in the specifications of dynamic relationships between service offerings that include complex expressions.

4 WSOL Expressions

WSOL 1.0 enabled specification of different types of expressions such as, Boolean expressions, arithmetic expressions, arithmeticWithUnit expressions, time expressions, string expressions, and quantified expressions, in WSOL constraints. Now, WSOL 1.1 also enables naming of these expressions and referencing them from other WSOL constraints. The main purpose behind naming and referencing of WSOL expressions is re-usability. A named WSOL expression could be defined only once in a WSOL constraint and then it could be re-used by referencing it from other constraints (i.e., constraints other than the one in which the named WSOL expression is defined). The need for naming and referencing of expressions was also discussed in [4].

4.1 Naming of all kinds of WSOL expressions (Boolean expressions, arithmetic expressions, arithmeticWithUnit expressions, time expressions, string expressions, and quantified expressions)

WSOL 1.1 enables naming of all kinds of expressions (i.e., Boolean expressions, arithmetic expressions, arithmeticWithUnit expressions, time expressions, string expressions, and quantified expressions) currently supported by the WSOL language. An attribute called *"name"* is added to the definitions of all kinds of WSOL expressions for this purpose. The use of attribute *"name"* is optional. So, naming of all kinds of WSOL expressions is optional. For illustration purposes, only an example of naming of Boolean expressions is shown here in Figure 13.

```
<wsol:serviceOffering name="SO1" service="buyStock:buyStockService">
  …
  …
  <wsol:constraint name="QoScons1" xsi:type="qosSchema:QoS" service="buyStock:buyStockService"
  portOrPortType="buyStock:buyStockServicePort" operation="buyStock:buySingleStockOperation">
    <expressionSchema:booleanExpression name="BE1">
      <expressionSchema:arithmeticWithUnitExpression>
        <expressionSchema:QoSmetric metricType="QoSMetricOntology:ResponseTime"
        metricUnit="QoSMeasOntology:millisecond" service="buyStock:buyStockService"
        portOrPortType="buyStock:buyStockServicePort"
        operation="buyStock:buySingleStockOperation" measuredBy="WSOL_INTERNAL"/>
      </expressionSchema:arithmeticWithUnitExpression>
      <expressionSchema:arithmeticWithUnitComparator type="&lt;"/>
      <expressionSchema:arithmeticWithUnitExpression>
        <wsol:numberWithUnitConstant>
          <wsol:number value="10"/>
          <wsol:unit type="QoSMeasOntology:millisecond"/>
        </wsol:numberWithUnitConstant>
      </expressionSchema:arithmeticWithUnitExpression>
    </expressionSchema:booleanExpression>
  </wsol:constraint>
  …
  …
</wsol:serviceOffering>
```

Figure 13 Example of naming of Boolean expressions

In the above example, a Boolean expression specified within a QoS constraint (*"QoScons1"*) is named *"BE1"* (shown in bold) with the help of the *"name"* attribute.

4.2 Referencing all kinds of WSOL expressions (Boolean expressions, arithmetic expressions, arithmeticWithUnit expressions, time expressions, string expressions, and quantified expressions) from other WSOL constraints

WSOL 1.1 enables referencing all kinds of expressions (i.e., Boolean expressions, arithmetic expressions, arithmeticWithUnit expressions, time expressions, string expressions, and quantified expressions) currently supported by the WSOL language from other WSOL constraints. For illustration purposes, only an example of referencing Boolean expressions from other constraints is shown here in Figure 14.

```
<wsol:serviceOffering name="SO5" service="buyStock:buyStockService">
  <wsol:constraint name="QoScons5" xsi:type="qosSchema:QoS" service="buyStock:buyStockService"
  portOrPortType="buyStock:buyStockServicePort" operation="buyStock:buySingleStockOperation">
    <expressionSchema:booleanExpression>
      <expressionSchema:booleanExpression>
        <expressionSchema:booleanExpressionRef name="tns:SO1.QoScons1.BE1"/>
      </expressionSchema:booleanExpression>
      <expressionSchema:binaryBooleanOperator type="AND"/>
      <expressionSchema:booleanExpression>
        <expressionSchema:arithmeticWithUnitExpression>
          <expressionSchema:QoSmetric metricType="QoSMetricOntology:ResponseTime"
          metricUnit="QoSMeasOntology:millisecond" service="buyStock:buyStockService"
          portOrPortType="buyStock:buyStockServicePort"
          operation="buyStock:buySingleStockOperation" measuredBy="WSOL_INTERNAL"/>
        </expressionSchema:arithmeticWithUnitExpression>
        <expressionSchema:arithmeticWithUnitComparator type=">"/>
        <expressionSchema:arithmeticWithUnitExpression>
          <wsol:numberWithUnitConstant>
            <wsol:number value="5"/>
            <wsol:unit type="QoSMeasOntology:millisecond"/>
          </wsol:numberWithUnitConstant>
        </expressionSchema:arithmeticWithUnitExpression>
      </expressionSchema:booleanExpression>
    </expressionSchema:booleanExpression>
  </wsol:constraint>
</wsol:serviceOffering>
```

Figure 14 Example of referencing Boolean expressions from other constraints

In the above example, a Boolean expression named *"BE1"*, specified within a QoS constraint *"QoScons1"* that is specified within a service offering *"SO1"* is referenced from other QoS constraint *"QoScons5"* that is specified within other service offering *"SO5"*.

5 WSOL Statements

5.1 Specification of existing types of WSOL statements (i.e., subscription, price default, price, penalty default, penalty, and management responsibility) using the general <statement> construct

The WSOL language enables specification of the following statements: the subscription statement, the price default statement, the price statement, the penalty default statement, the penalty statement, and the management responsibility statement. WSOL 1.0 enabled specification of each of these existing types of WSOL statements using a specific element defined for that type of statement (i.e., WSOL 1.0 enabled specification of a price statement using a <price> element, a subscription statement using a <subscription> element, and so on). In addition, there was an extensibility <statement> element for new types of WSOL statements that might be defined in future. But, such specification of existing types of WSOL statements did not seem to be a consistent and flexible solution. So, the WSOL language was modified. Now, WSOL 1.1 enables specification of each of these existing types of WSOL statements using the general <*statement*> element (already defined in the WSOL language). For backward compatibility reasons, WSOL 1.1 will also enable specification of existing types of WSOL statements according to WSOL 1.0 for some time.

Note that WSOL 1.1 both contains new grammar for subscription statements and enables specification of subscription information through new attributes of the <serviceOffering> element, discussed in Section 7 of this report. Either one can be used, but we recommend the use of the new attributes of the <serviceOffering> element. (Maybe the subscription statement will not be supported in future versions of WSOL.) If more than one subscription information is specified within a service offering, the subscription amounts are added. Figure 15 shows an example of specification of subscription statement according to WSOL 1.1 using the general <*statement*> construct.

```
<wsol:statement name="subscriptionStmt1" xsi:type="subscriptionSchema:subscription">
   <wsol:numberWithUnitConstant>
      <wsol:number value="50"/>
      <wsol:unit type="currencyOntology:Dollar"/>
   </wsol:numberWithUnitConstant>
   <wsol:subscriptionDuration duration="P1Y"/>
</wsol:statement>
```

Figure 15 Example of specification of subscription statement according to WSOL 1.1 using the general <statement> construct

The "*xsi:type*" attribute in the general <*statement*> element is used to specify the type of statement. Its value is different for different types of WSOL statements.

In the above example, a subscription statement named *"subscriptionStmt1"* is specified using the general <*statement*> element. The value of the "*xsi:type*" attribute in this statement is

"subscriptionSchema:subscription" which denotes that the corresponding statement is a subscription statement. Also, the value *"subscriptionSchema:subscription"* means that the grammar for a subscription statement is defined in its corresponding schema (i.e., *"subscriptionSchema"*), shown in Figure 16 and Figure 17. The example subscription statement states that the subscription price to be paid by a Web Service consumer to the Web Service supplier is 50 Dollars and the subscription duration is 1 year.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/SubscriptionSchema"
xmlns:subscriptionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/SubscriptionSchema"
xmlns:wsol="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/SubscriptionSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
   schemaLocation="Schemas/ExpressionSchema.xsd"/>
   <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
   schemaLocation="Schemas/WSOLSchema.xsd"/>
   <xsd:complexType name="subscription">
      <xsd:complexContent>
         <xsd:extension base="wsol:statementType">
            <xsd:sequence>
               <xsd:element ref="wsol:numberWithUnitConstant"/>
               <xsd:element ref="wsol:subscriptionDuration"/>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
</xsd:schema>
```

Figure 16 Subscription Schema

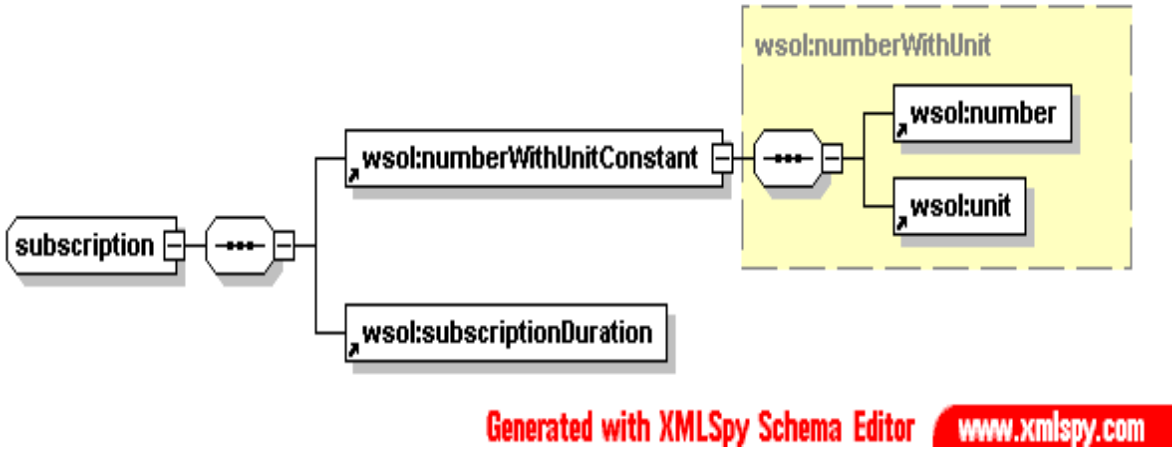


Figure 17 Subscription Schema (Graphical Representation)

Figure 18 shows an example of specification of price default statement according to WSOL 1.1 using the general <*statement*> construct.

```
<wsol:statement name="priceDefaultStmt1" xsi:type="priceDefaultSchema:priceDefault">
   <wsol:numberWithUnitConstant>
      <wsol:number value="2"/>
      <wsol:unit type="currencyOntology:Dollar"/>
   </wsol:numberWithUnitConstant>
</wsol:statement>
```

Figure 18 Example of specification of price default statement according to WSOL 1.1 using the general <statement> construct

In the above example, a price default statement named *"priceDefaultStmt1"* is specified using the general <*statement*> element. The value of the "*xsi:type*" attribute in this statement is *"priceDefaultSchema:priceDefault"* which denotes that the corresponding statement is a price default statement and that the grammar is defined in *"priceDefaultSchema"* (Figure 19 and Figure 20). The example price default statement states that the default price to be paid by a Web Service consumer to the Web Service supplier is 2 Dollars.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/PriceDefaultSchema"
xmlns:priceDefaultSchema="http://www.sce.carleton.ca/~kpatel/WSOL/PriceDefaultSchema"
xmlns:wsol="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/PriceDefaultSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
   schemaLocation="Schemas/ExpressionSchema.xsd"/>
   <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
   schemaLocation="Schemas/WSOLSchema.xsd"/>
   <xsd:complexType name="priceDefault">
      <xsd:complexContent>
         <xsd:extension base="wsol:statementType">
            <xsd:sequence>
               <xsd:element ref="wsol:numberWithUnitConstant"/>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
</xsd:schema>
```

Figure 19 Price Default Schema

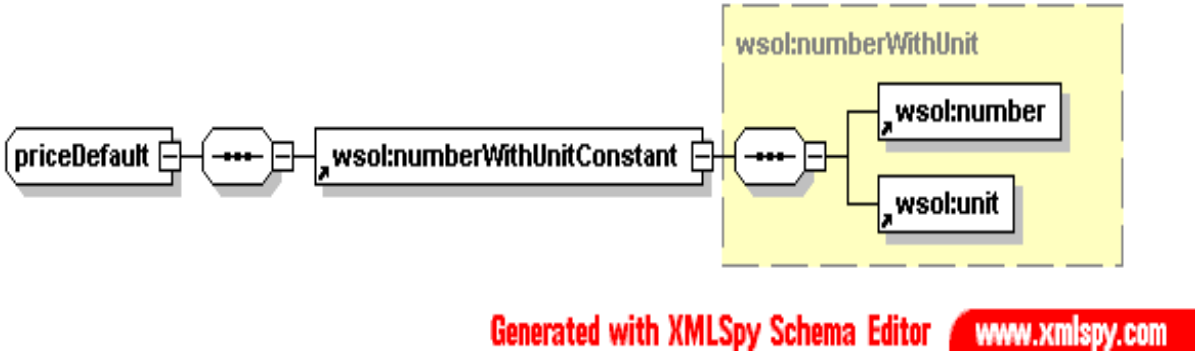


Figure 20 Price Default Schema (Graphical Representation)

Figure 21 shows an example of specification of price statement according to WSOL 1.1 using the general <*statement*> construct.

```
<wsol:statement name="priceStmt1" xsi:type="priceSchema:price" service="buyStock:buyStockService"
portOrPortType="buyStock:buyStockServicePort" operation="buyStock:buySingleStockOperation">
   <wsol:numberWithUnitConstant>
      <wsol:number value="5"/>
      <wsol:unit type="currencyOntology:Dollar"/>
   </wsol:numberWithUnitConstant>
</wsol:statement>
```

Figure 21 Example of specification of price statement according to WSOL 1.1 using the general <statement> construct

In the above example, a price statement named *"priceStmt1"* is specified using the general *<statement>* element. The "*xsi:type*" attribute in this statement has a value of *"priceSchema:price"* which denotes that the corresponding statement is a price statement and that the grammar is defined in *"priceSchema"* (Figure 22 and Figure 23). The example price statement is defined for a particular operation (*"buySingleStockOperation"*) and for a particular port (*"buyStockServicePort"*) of a particular Web Service (*"buyStockService"*) and it states that the price to be paid by a Web Service consumer to the Web Service supplier is 5 Dollars.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/PriceSchema"
xmlns:priceSchema="http://www.sce.carleton.ca/~kpatel/WSOL/PriceSchema"
xmlns:wsol="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/PriceSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
   schemaLocation="Schemas/ExpressionSchema.xsd"/>
   <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
   schemaLocation="Schemas/WSOLSchema.xsd"/>
   <xsd:complexType name="price">
      <xsd:complexContent>
         <xsd:extension base="wsol:statementType">
            <xsd:sequence>
               <xsd:element ref="wsol:numberWithUnitConstant"/>
            </xsd:sequence>
            <xsd:attribute name="service" type="xsd:QName" use="required"/>
            <xsd:attribute name="portOrPortType" type="xsd:QName" use="required"/>
            <xsd:attribute name="operation" type="xsd:QName" use="required"/>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
</xsd:schema>
```

Figure 22 Price Schema

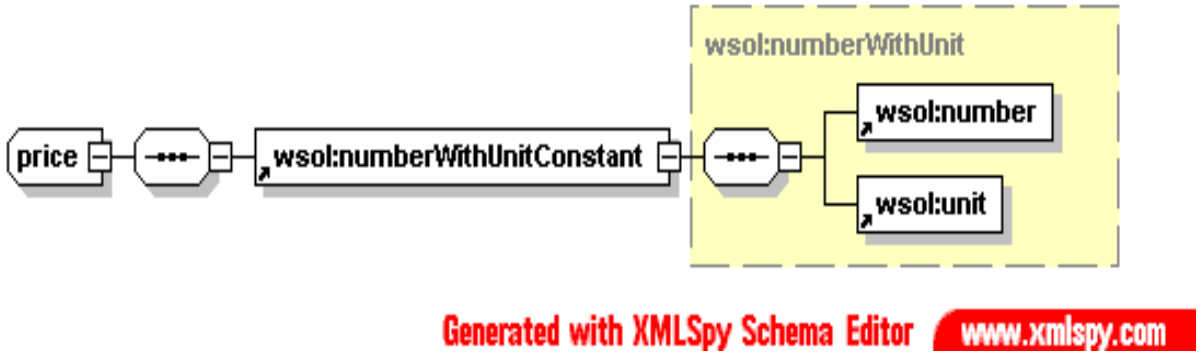


Figure 23 Price Schema (Graphical Representation)

Figure 24 shows an example of specification of penalty default statement according to WSOL 1.1 using the general <*statement*> construct.

```
<wsol:statement name="penaltyDefaultStmt1" xsi:type="penaltyDefaultSchema:penaltyDefault">
   <wsol:numberWithUnitConstant>
      <wsol:number value="-1"/>
      <wsol:unit type="currencyOntology:Dollar"/>
   </wsol:numberWithUnitConstant>
</wsol:statement>
```

Figure 24 Example of specification of penalty default statement according to WSOL 1.1 using the general <statement> construct

In the above example, a penalty default statement named *"penaltyDefaultStmt1"* is specified using the general <*statement*> element. The value of the "*xsi:type*" attribute in this statement is *"penaltyDefaultSchema:penaltyDefault"* which denotes that the corresponding statement is a penalty default statement and that the grammar is defined in *"penaltyDefaultSchema"* (Figure 25 and Figure 26). The example penalty default statement states that the default penalty to be paid by the Web Service supplier to a Web Service consumer is 1 Dollar.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/PenaltyDefaultSchema"
xmlns:penaltyDefaultSchema="http://www.sce.carleton.ca/~kpatel/WSOL/PenaltyDefaultSchema"
xmlns:wsol="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/PenaltyDefaultSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
   schemaLocation="Schemas/ExpressionSchema.xsd"/>
   <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
   schemaLocation="Schemas/WSOLSchema.xsd"/>
   <xsd:complexType name="penaltyDefault">
      <xsd:complexContent>
         <xsd:extension base="wsol:statementType">
            <xsd:sequence>
               <xsd:element ref="wsol:numberWithUnitConstant"/>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
</xsd:schema>
```

Figure 25 Penalty Default Schema

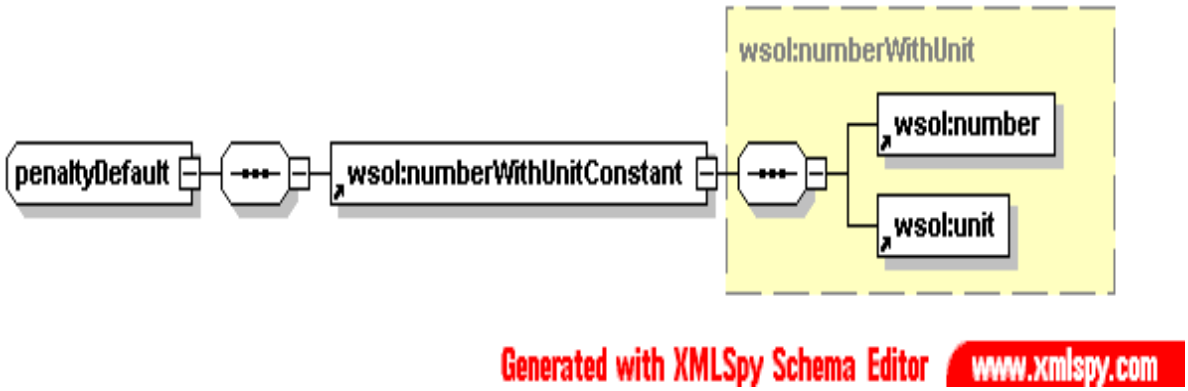


Figure 26 Penalty Default Schema (Graphical Representation)

Figure 27 shows an example of specification of penalty statement according to WSOL 1.1 using the general <*statement*> construct.

```
<wsol:statement name="penaltyStmt1" xsi:type="penaltySchema:penalty"
service="buyStock:buyStockService" portOrPortType="buyStock:buyStockServicePort"
operation="buyStock:buySingleStockOperation">
   <wsol:numberWithUnitConstant>
      <wsol:number value="-2.5"/>
      <wsol:unit type="currencyOntology:Dollar"/>
   </wsol:numberWithUnitConstant>
</wsol:statement>
```

Figure 27 Example of specification of penalty statement according to WSOL 1.1 using the general <statement> construct

In the above example, a penalty statement named *"penaltyStmt1"* is specified using the general *<statement>* element. The value of the "*xsi:type*" attribute in this statement is *"penaltySchema:penalty"* which denotes that the corresponding statement is a penalty statement and that the grammar is defined in *"penaltySchema"* (Figure 28 and Figure 29). The example penalty statement is defined for a particular operation (*"buySingleStockOperation"*) and for a particular port (*"buyStockServicePort"*) of a particular Web Service (*"buyStockService"*) and it states that the penalty to be paid by the Web Service supplier to a Web Service consumer is 2.5 Dollars.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/PenaltySchema"
xmlns:penaltySchema="http://www.sce.carleton.ca/~kpatel/WSOL/PenaltySchema"
xmlns:wsol="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/PenaltySchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
   schemaLocation="Schemas/ExpressionSchema.xsd"/>
   <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
   schemaLocation="Schemas/WSOLSchema.xsd"/>
   <xsd:complexType name="penalty">
      <xsd:complexContent>
         <xsd:extension base="wsol:statementType">
            <xsd:sequence>
               <xsd:element ref="wsol:numberWithUnitConstant"/>
            </xsd:sequence>
            <xsd:attribute name="service" type="xsd:QName" use="required"/>
            <xsd:attribute name="portOrPortType" type="xsd:QName" use="required"/>
            <xsd:attribute name="operation" type="xsd:QName" use="required"/>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
</xsd:schema>
```

Figure 28 Penalty Schema

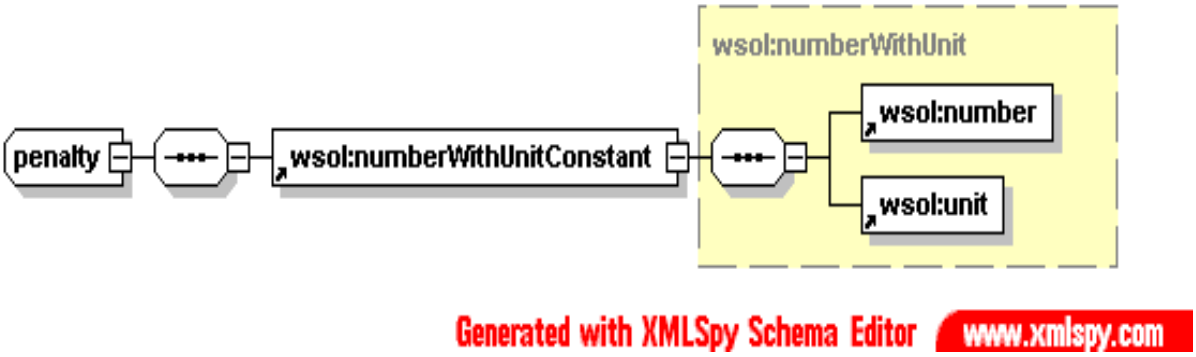


Figure 29 Penalty Schema (Graphical Representation)

Figure 30 shows an example of specification of management responsibility statement according to WSOL 1.1 using the general <*statement*> construct.

```
<wsol:statement name="mgmtRespStmtSO1"
xsi:type="managementResponsibilitySchema:managementResponsibility">
   <managementResponsibilitySchema:supplierResponsibility scope="tns:SO1.AccRghtCons2"/>
   <managementResponsibilitySchema:consumerResponsibility scope="tns:SO1.C3"/>
</wsol:statement>
```

Figure 30 Example of specification of management responsibility statement according to WSOL 1.1 using the general <statement> construct

In the above example, a management responsibility statement named *"mgmtRespStmtSO1"* is specified using the general <*statement*> element. The value of the "*xsi:type*" attribute in this statement is *"managementResponsibilitySchema:managementResponsibility"* which denotes that the corresponding statement is a management responsibility statement and that the grammar is defined in *"managementResponsibilitySchema"* (Figure 31 and Figure 32). The example management responsibility statement states that the Web Service supplier has the management responsibility for checking an access right constraint named *"AccRghtCons2"* specified in a service offering *"SO1"*. It also states that the Web Service consumer has the management responsibility for checking a functional constraint named *"C3"* also specified in the service offering *"SO1"*.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
targetNamespace=”http://www.sce.carleton.ca/~kpatel/WSOL/ManagementResponsibilitySchema “
xmlns:managementResponsibilitySchema="http://www.sce.carleton.ca/~kpatel/WSOL/ManagementRespon
sibilitySchema" xmlns:wsol="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/ManagementResponsibilitySchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
   schemaLocation="Schemas/ExpressionSchema.xsd"/>
   <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
   schemaLocation="Schemas/WSOLSchema.xsd"/>
   <xsd:complexType name="managementResponsibility">
      <xsd:complexContent>
         <xsd:extension base="wsol:statementType">
            <xsd:sequence>
               <xsd:element ref="supplierResponsibility" minOccurs="0" maxOccurs="unbounded"/>
               <xsd:element ref="consumerResponsibility" minOccurs="0" maxOccurs="unbounded"/>
               <xsd:element ref="independentResponsibility" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
   <xsd:element name="supplierResponsibility">
      <xsd:complexType>
         <xsd:attribute name="scope" type="xsd:QName" use="required"/>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="consumerResponsibility">
      <xsd:complexType>
         <xsd:attribute name="scope" type="xsd:QName" use="required"/>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="independentResponsibility">
      <xsd:complexType>
         <xsd:attribute name="scope" type="xsd:QName" use="required"/>
         <xsd:attribute name="entity" type="xsd:anyURI" use="required"/>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

Figure 31 Management Responsibility Schema

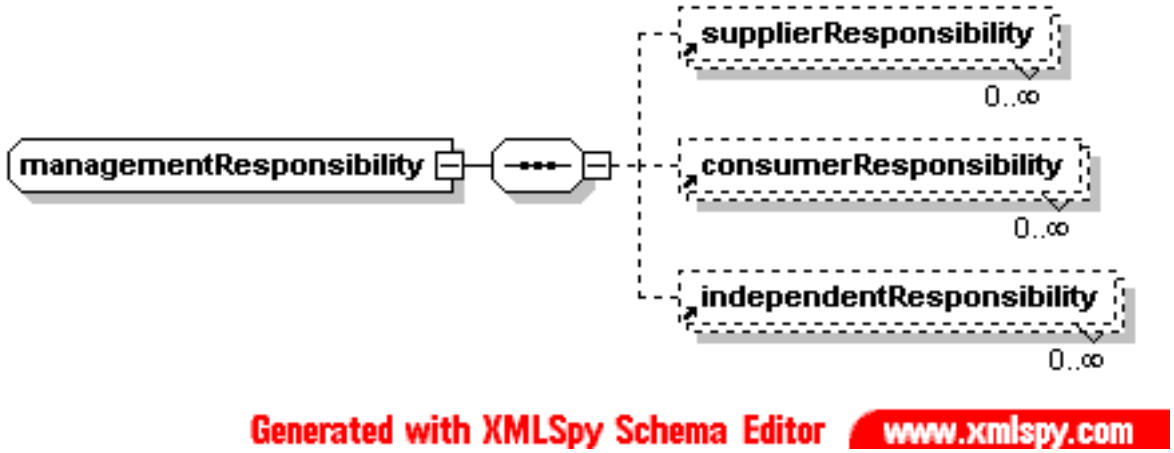


Figure 32 Management Responsibility Schema (Graphical Representation)

The "Premier" WSOL parser has been improved to validate the existing types of WSOL statements now specified using the general <*statement*> construct. While validating, the parser stores important data from WSOL statements into a symbol table and also detects and reports syntax as well as semantic errors in them. For backward compatibility reasons, the "Premier" WSOL parser will continue to support for some time the validation of existing types of WSOL statements specified according to WSOL 1.0.

5.2 Specification of Additional Penalty Statements

WSOL 1.1 enables specification of additional penalty statements. These statements are specified in addition to the general penalty statements. A general penalty statement specifies the penalty to be paid by a Web Service supplier to the Web Service consumer for not being able to fulfill all the constraints specified for a particular operation being used by the Web Service consumer. (In other words: "for not being able to fulfil at least one, any one, of the constraints for this operation".) An additional penalty statement enables specification of additional penalty, to be paid in addition to the general penalty, by a Web Service supplier to the Web Service consumer for not being able to fulfill a particular constraint specified for a particular operation being used by the Web Service consumer. This is useful because fulfillment of some constraints might be more important to the consumer than fulfillment of other constraints and, thus, should correspond to higher penalties.

Figure 33 shows an example of specification of additional penalty statement according to WSOL 1.1 using the general <*statement*> construct.

```
<wsol:statement name="additionalPenaltyStmt1" xsi:type="additionalPenaltySchema:additionalPenalty"
constraint="tns:SO1.QoScons1">
   <wsol:numberWithUnitConstant>
      <wsol:number value="-2"/>
      <wsol:unit type="currencyOntology:Dollar"/>
   </wsol:numberWithUnitConstant>
</wsol:statement>
```

Figure 33 Example of specification of additional penalty statement according to WSOL 1.1 using the general <statement> construct

In the above example, an additional penalty statement named *"additionalPenaltyStmt1"* is specified using the general <*statement*> element. The value of the "*xsi:type*" attribute in this statement is *"additionalPenaltySchema: additionalPenalty"* which denotes that the corresponding statement is an additional penalty statement and that the grammar is defined in *"additionalPenaltySchema"* (Figure 34 and Figure 35). The example additional penalty statement is defined for a particular constraint ("*tns:SO1.QoScons1*") specified within a particular Service Offering ("*tns:SO1*") that is specified for a particular Web Service ("*buyStockService"*). This example additional penalty statement states that the additional penalty to be paid by the Web Service supplier to a Web Service consumer for not being able to satisfy the constraint ("*tns:SO1.QoScons1"*) is 2 Dollars. Similarly to penalty default and penalty statements, the monetary value specified inside an additional penalty statement must be a negative number.

```
<?xml version="1.0"?>
<xsd:schema targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/AdditionalPenaltySchema"
xmlns:additionalPenaltySchema="http://www.sce.carleton.ca/~kpatel/WSOL/AdditionalPenaltySchema"
xmlns:wsol="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/AdditionalPenaltySchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
   schemaLocation="Schemas/ExpressionSchema.xsd"/>
   <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
   schemaLocation="Schemas/WSOLSchema.xsd"/>
   <xsd:complexType name="additionalPenalty">
      <xsd:complexContent>
         <xsd:extension base="wsol:statementType">
            <xsd:sequence>
               <xsd:element ref="wsol:numberWithUnitConstant"/>
            </xsd:sequence>
            <xsd:attribute name="constraint" type="xsd:QName" use="required"/>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
</xsd:schema>
```

Figure 34 Additional Penalty Schema

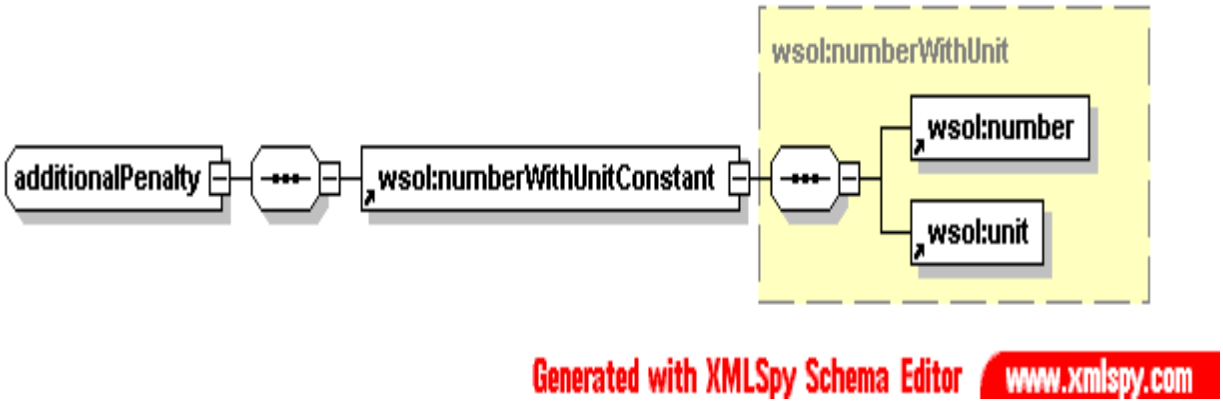


Figure 35 Additional Penalty Schema (Graphical Representation)

The “Premier” WSOL parser has been improved to validate the WSOL additional penalty statements. While validating, the WSOL parser stores important data (such as, the name of constraint for which the additional penalty statement is specified, and the monetary value and currency of additional penalty) into a symbol table. The WSOL parser also detects and reports syntax as well as semantic errors in the WSOL additional penalty statements. A few examples of such errors are given below:

- the value of "*xsi:type*" attribute, that denotes the type of statement, is not *"additionalPenaltySchema: additionalPenalty"*,
- the attribute *"constraint"* is not specified,
- the value of attribute *"constraint"*, which denotes the name of constraint for which the additional penalty statement is specified, is not specified,
- the value of attribute *"constraint"* (which denotes the name of constraint for which the additional penalty statement is specified) is specified but is incorrect,
- the monetary value and currency of additional penalty are not specified,
- the monetary value of additional penalty is specified but is a positive number (i.e., greater that 0), and
- the monetary currency of additional penalty is specified but is incorrect.

5.3 Specification of WSOL Service Offering Validity Statements

WSOL 1.1 enables specification of validity period for WSOL Service Offerings with the help of a special type of statement called "WSOL Service Offering Validity Statement". A Service Offering Validity statement enables specification of the validity duration and the validity expiration date and time of a particular WSOL SO. The validity duration specified for a particular WSOL SO means that the WSOL SO is valid only for the duration that is specified (the validity duration is evaluated from the time when a Web Service consumer starts using the WSOL SO). After the validity duration has expired the particular WSOL SO will no longer be valid. The validity expiration date and time specified for a particular WSOL SO means that the WSOL SO is valid only until the date and time specified and will be no longer valid after the expiration date and time even though the validity duration has not expired. Similarly, if the validity duration for a particular WSOL SO has expired but the validity expiration date and time have not expired then also the WSOL SO is no longer valid. If no WSOL SO validity statement is specified for a WSOL SO, then it is to be assumed that that particular WSOL SO is valid forever.

Note that the same validity period information can now also be specified through new attributes of the <serviceOffering> element. While both approaches can be used, we recommend using the attributes of the <serviceOffering> element. (Maybe the specification of validity periods through statements will not be supported in future versions of WSOL.)

Figure 36 shows three examples of specification of WSOL Service Offering Validity Statements according to WSOL 1.1 using the general <*statement*> construct.

```
<wsol:statement name="soValidityStmt1" xsi:type="soValiditySchema:soValidity">
   <soValiditySchema:soName name="tns:SO1"/>
   <soValiditySchema:soValidityDuration duration="P2Y"/>
   <soValiditySchema:soValidityExpiration expDateTime="2010-12-31T00:00:00-05:00"/>
</wsol:statement>

<wsol:statement name="soValidityStmt2" xsi:type="soValiditySchema:soValidity">
   <soValiditySchema:soName name="tns:SO2"/>
   <soValiditySchema:soValidityDuration duration="P1Y"/>
   <soValiditySchema:soValidityExpiration expDateTime="2008-12-31T00:00:00-05:00"/>
</wsol:statement>

<wsol:statement name="soValidityStmt3" xsi:type="soValiditySchema:soValidity">
   <soValiditySchema:soName name="tns:SO3"/>
   <soValiditySchema:soValidityDuration duration="P1Y6M"/>
   <soValiditySchema:soValidityExpiration expDateTime="2005-12-31T00:00:00-05:00"/>
</wsol:statement>
```

Figure 36 Examples of specification of WSOL Service Offering Validity Statements

The first example WSOL Service Offering Validity statement named *"soValidityStmt1"* states that the Service Offering *"SO1"* is valid for a duration of *"P2Y"*, i.e., for 2 years, and will expire on *"2010-12-31T00:00:00-05:00"* date and time, i.e., on 31st December 2010 at midnight. Similarly, the second example WSOL Service Offering Validity statement named *"soValidityStmt2"* states that the Service Offering *"SO2"* is valid for a duration of *"P1Y"*, i.e., for 1 year, and will expire on *"2008-12-31T00:00:00-05:00"* date and time, i.e., on 31st December 2008 at midnight. The third example WSOL Service Offering Validity statement named *"soValidityStmt3"* states that the Service Offering *"SO3"* is valid for a duration of *"P1Y6M"*, i.e., for 1 year and 6 months, and will expire on *"2005-12-31T00:00:00-05:00"* date and time, i.e., on 31st December 2005 at midnight.

The XML grammar that enables specification of a WSOL Service Offering Validity Statement is defined in a schema called *"SOValiditySchema.xsd"* (Figure 37 and Figure 38).

```
<?xml version="1.0"?>
<xsd:schema targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/SOValiditySchema"
xmlns:soValiditySchema="http://www.sce.carleton.ca/~kpatel/WSOL/SOValiditySchema"
xmlns:wsol="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/SOValiditySchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
   schemaLocation="Schemas/ExpressionSchema.xsd"/>
   <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
   schemaLocation="Schemas/WSOLSchema.xsd"/>
   <xsd:complexType name="soValidity">
      <xsd:complexContent>
         <xsd:extension base="wsol:statementType">
            <xsd:sequence>
               <xsd:element ref="soName" maxOccurs="unbounded"/>
               <xsd:element ref="soValidityDuration"/>
               <xsd:element ref="soValidityExpiration"/>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
   <xsd:element name="soName">
      <xsd:complexType>
         <xsd:attribute name="name" type="xsd:QName" use="required"/>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="soValidityDuration">
      <xsd:complexType>
         <xsd:attribute name="duration" type="xsd:duration" use="required"/>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="soValidityExpiration">
      <xsd:complexType>
         <xsd:attribute name="expDateTime" type="xsd:dateTime" use="required"/>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

Figure 37 Service Offering Validity Schema (Grammar for the specification of a WSOL Service Offering Validity Statement)

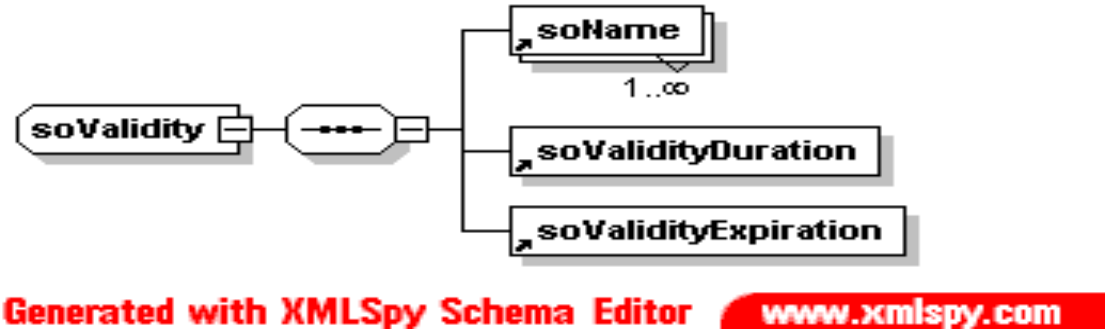


Figure 38 Service Offering Validity Schema (Graphical Representation)

The "Premier" WSOL parser has been improved to validate the WSOL service offering validity statements. While validating, the WSOL parser stores important data (such as, the name of the service offering for which the WSOL service offering validity statement is specified, the validity duration specified for the particular service offering, and the validity expiration date and time specified for the particular service offering) into a symbol table. The WSOL parser also detects and reports syntax as well as semantic errors in the WSOL service offering validity statements. A few examples of such errors are given below:

- the value of "*xsi:type*" attribute, that denotes the type of statement, is not *"soValiditySchema:soValidity"*,
- the name of the service offering, for which the WSOL service offering validity statement is defined, is not specified,
- the name of the service offering, for which the WSOL service offering validity statement is defined, is specified but is incorrect,
- the validity duration for the service offering is not specified, and
- the validity expiration date and time for the service offering are not specified.

6 WSOL Reusability Constructs

6.1 Specification of CGs and CGTs in which only some constraints have to be satisfied

WSOL 1.0 enabled only specification of constraint groups (CGs) and constraint group templates (CGTs) in which always all constraints have to be satisfied. The new version of WSOL language, i.e., WSOL 1.1, enables specification of CGs and CGTs in which either all or some constraints have to be satisfied. In order to enable such specification, the definitions of CGs and CGTs in WSOL 1.1 contain an additional attribute called "*satisfiedConstraints*". The value of attribute "*satisfiedConstraints*" can be "ALL", "ONEORMORE", "EXACTLYONE", or "NONE".

If the value of the attribute "*satisfiedConstraints*" is "ALL" (which is the default value), it means that all the constraints from the given CG or CGT have to be satisfied. If the value of the attribute "*satisfiedConstraints*" is "ONEORMORE", it means that at least one constraint from the given CG or CGT has to be satisfied. If the value of the attribute "*satisfiedConstraints*" is "EXACTLYONE", it means that exactly one constraint from the given CG or CGT has to be satisfied. And, if the value of the attribute "*satisfiedConstraints*" is "NONE", it means that all constraints from the given CG or CGT have to be unsatisfied. (Currently, we do not see any use of the attribute value "NONE", but still it has been provided in WSOL for eventual future use.) The specification of the attribute "*satisfiedConstraints*" is made optional. If the attribute is not specified in a CG or CGT specification, then it is assumed that all the constraints from the given CG or CGT have to be satisfied, since, the default value of attribute "*satisfiedConstraints*" is "ALL".

The attribute *"satisfiedConstraints"* and its value specified in a CG or CGT applies only to the constraints and CGs specified within the CG or CGT. It does not apply to the statements specified within the CG or CGT because all the statements specified within a CG or CGT are always valid. For example, if a CG with the value of attribute "*satisfiedConstraints*" equal to "ONEORMORE" has two penalty statements for the same operation, then the result penalty to be paid is the sum of these two penalties. (The same would be the case with any other value of the attribute "*satisfiedConstraints*".) Also, an important rule for the attribute *"satisfiedConstraints"* is that its value in the extending and the extended CG or CGT must always be the same. Figure 39 shows the new grammar for the specification of a WSOL Constraint Group (CG). The parts, shown in bold, are the newly added parts. Figure 40 shows the graphical representation of the new grammar for the specification of a WSOL Constraint Group (CG).

The need for extending WSOL with the specification of CGs and CGTs in which only some constraints have to be specified was discussed in [4].

```
<xsd:element name="CG" type="wsol:CGType"/>
  <xsd:complexType name="CGType">
    <xsd:sequence>
      <xsd:element ref="wsol:externalOperationCall" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:constraint" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:subscription" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:price" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:priceDefault" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:penalty" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:penaltyDefault" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:managementResponsibility" minOccurs="0"/>
      <xsd:element ref="wsol:statement" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:include" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:CG" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:instantiate" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:NCName" use="required"/>
    <xsd:attribute name="extends" type="xsd:QName" use="optional" default="WSOL-NONE"/>
    <xsd:attribute name="service" type="xsd:QName" use="required"/>
    <xsd:attribute name="portOrPortType" type="xsd:QName" use="required"/>
    <xsd:attribute name="operation" type="xsd:QName" use="required"/>
    <xsd:attribute name="satisfiedConstraints" type="wsol:satisfiedAttributeValues"
    use="optional" default="ALL"/>
  </xsd:complexType>
  <xsd:simpleType name="satisfiedAttributeValues">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="ALL"/>
      <xsd:enumeration value="ONEORMORE"/>
      <xsd:enumeration value="EXACTLYONE"/>
      <xsd:enumeration value="NONE"/>
    </xsd:restriction>
  </xsd:simpleType>
```

Figure 39 New grammar for the specification of a WSOL Constraint Group (CG)

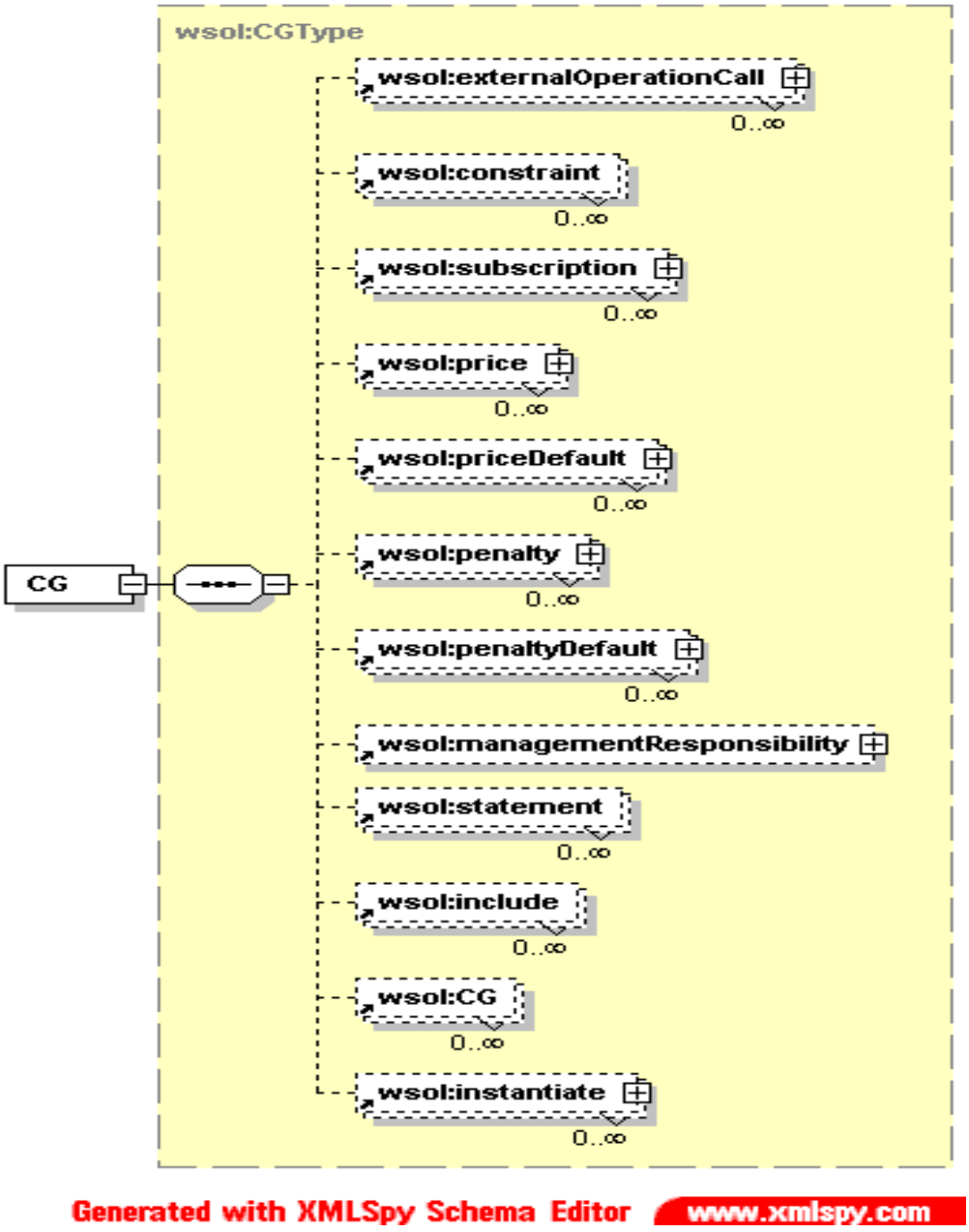


Figure 40 Graphical representation of the new grammar for the specification of a WSOL Constraint Group (CG)

An example of a constraint group (CG) specified using the improved WSOL CG grammar is given in Figure 41:

```
<wsol:CG name="CG6" service="buyStock:buyStockService"
portOrPortType="buyStock:buyStockServicePort" operation="buyStock:buySingleStockOperation"
satisfiedConstraints="EXACTLYONE">
  <wsol:constraint name="C4" xsi:type="preConditionSchema:preCondition"
  service="buyStock:buyStockService" portOrPortType="buyStock:buyStockServicePort"
  operation="buyStock:buySingleStockOperation">
    <expressionSchema:booleanExpression>
      <expressionSchema:arithmeticExpression>
        <expressionSchema:arithmeticVariable avName="buyStock:buySingleStockRequest.maxPrice"/>
      </expressionSchema:arithmeticExpression>
      <expressionSchema:arithmeticComparator type=">"/>
      <expressionSchema:arithmeticExpression>
        <expressionSchema:arithmeticConstant>
          <expressionSchema:floatConstant value="0.0"/>
        </expressionSchema:arithmeticConstant>
      </expressionSchema:arithmeticExpression>
    </expressionSchema:booleanExpression>
  </wsol:constraint>
  <wsol:statement name="subscriptionStmt2" xsi:type="subscriptionSchema:subscription">
    <wsol:numberWithUnitConstant>
      <wsol:number value="20"/>
      <wsol:unit type="currencyOntology:Dollar"/>
    </wsol:numberWithUnitConstant>
    <wsol:subscriptionDuration duration="P1Y"/>
  </wsol:statement>
  …
  <wsol:include constructName="tns:SO1.C3" resService="buyStock:buyStockService"
  resPortOrPortType="buyStock:buyStockServicePort"
  resOperation="buyStock:buySingleStockOperation" resName="C40"/>
</wsol:CG>
```

Figure 41 Example of a constraint group (CG) specified using the improved WSOL constraint group (CG) grammar

In the above example, a constraint group named *"CG6"* is defined for a particular service (*"buyStockService"*), port (*"buyStockServicePort"*), and operation (*"buySingleStockOperation"*). The attribute "*satisfiedConstraints*" is specified in the CG definition with a value of "EXACTLYONE". The CG contains the definition of a constraint named *"C4"* and also includes a constraint named *"C3"* from service offering *"SO1"*. Since the attribute "*satisfiedConstraints*" has a value of "EXACTLYONE", either the constraint *"C4"* or the included constraint *"C3"* has to be satisfied within this CG (but not both of them).

Figure 42 shows the new grammar for the specification of a WSOL Constraint Group Template (CGT). The parts, shown in bold, are the newly added parts. Figure 43 shows the graphical representation of the new grammar for the specification of a WSOL Constraint Group Template (CGT).

```
<xsd:element name="CGT" type="wsol:CGTType"/>
  <xsd:complexType name="CGTType">
    <xsd:sequence>
      <xsd:element ref="wsol:parameter" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:externalOperationCall" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:constraint" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:subscription" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:price" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:priceDefault" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:penalty" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:penaltyDefault" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:managementResponsibility" minOccurs="0"/>
      <xsd:element ref="wsol:statement" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:include" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:CG" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:instantiate" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:NCName" use="required"/>
    <xsd:attribute name="extends" type="xsd:QName" default="WSOL-NONE"/>
    <xsd:attribute name="service" type="xsd:QName" use="required"/>
    <xsd:attribute name="portOrPortType" type="xsd:QName" use="required"/>
    <xsd:attribute name="operation" type="xsd:QName" use="required"/>
    <xsd:attribute name="satisfiedConstraints" type="wsol:satisfiedAttributeValues"
    use="optional" default="ALL"/>
  </xsd:complexType>
  <xsd:element name="parameter" type="wsol:parameterType"/>
  <xsd:complexType name="parameterType">
    <xsd:attribute name="name" type="xsd:NCName" use="required"/>
    <xsd:attribute name="dataType" type="xsd:QName" use="required"/>
    <xsd:attribute name="unit" type="xsd:QName" default="WSOL-NOUNIT"/>
  </xsd:complexType>
```

Figure 42 New grammar for the specification of a WSOL Constraint Group Template (CGT)

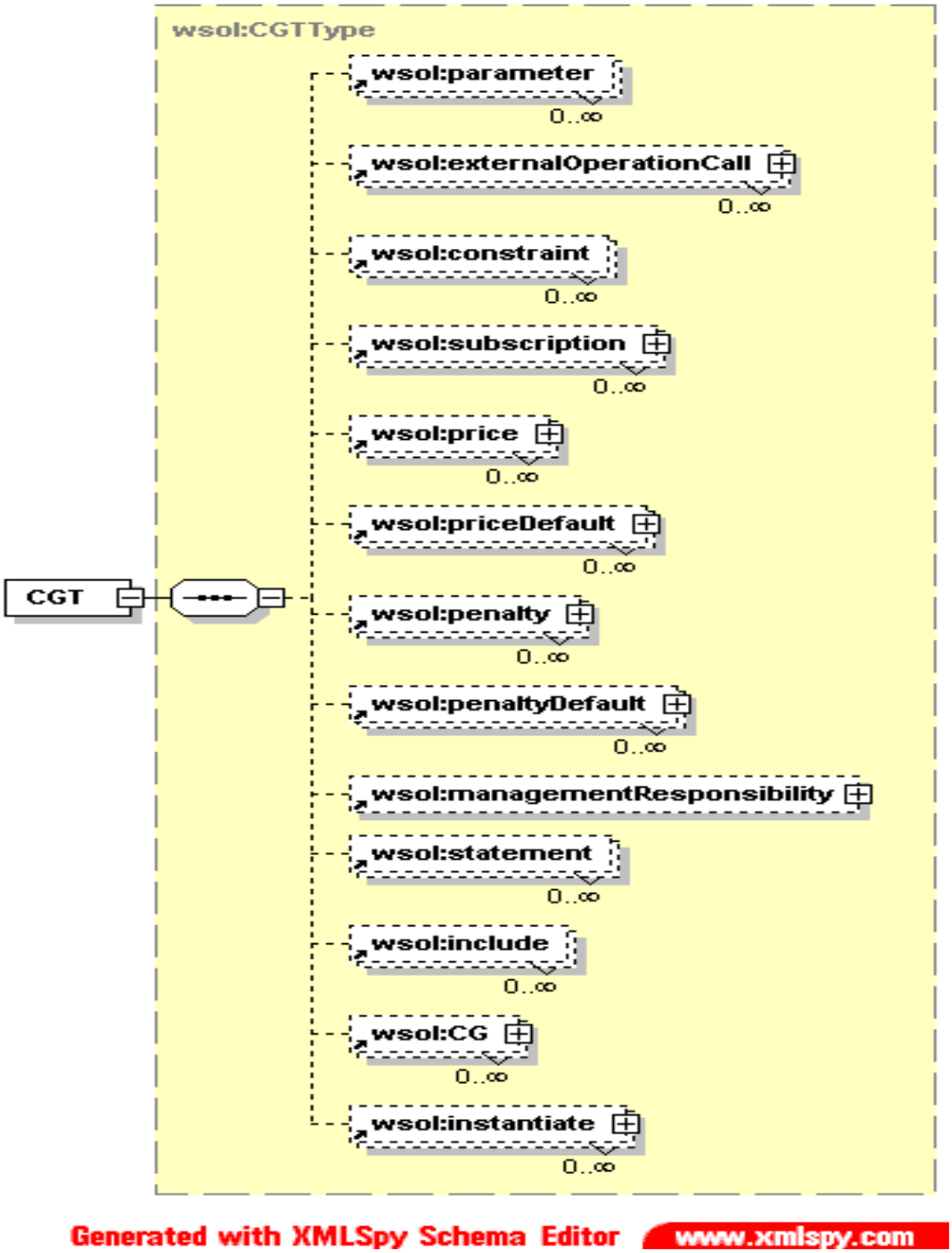


Figure 43 Graphical representation of the new grammar for the specification of a WSOL Constraint Group Template (CGT)

An example of a constraint group template (CGT) specified using the improved WSOL CGT grammar, is given in Figure 44:

```
<wsol:CGT name="CGT2" service="buyStock:buyStockService"
portOrPortType="buyStock:buyStockServicePort" operation="buyStock:buySingleStockOperation"
satisfiedConstraints=”ONEORMORE”>
   <wsol:parameter name="responseTime" dataType="wsol:numberWithUnit"
   unit="QoSMeasOntology:millisecond"/>
   <wsol:externalOperationCall callID="extOp7"
   calledOperDescription="timeLookup:timeLookupPortType.currentDateTime"
   calledOperImplementation="timeLookup:timeLookupService.timeLookupServicePort"
   service="buyStock:buyStockService" portOrPortType="buyStock:buyStockServicePort"
   operation="buyStock:buySingleStockOperation"/>
   <wsol:constraint name="C15" xsi:type="postConditionSchema:postCondition"
   service="buyStock:buyStockService" portOrPortType="buyStock:buyStockServicePort"
   operation="buyStock:buySingleStockOperation">
      <expressionSchema:booleanExpression>
         <expressionSchema:timeExpression>
            <expressionSchema:externalOperationResult callID="tns:CGT2.extOp7"
            resultPartName="timeService:currentDateTimeResponse.currentDateTime"/>
         </expressionSchema:timeExpression>
         <expressionSchema:timeComparator type="=="/>
         <expressionSchema:timeExpression>
            <expressionSchema:timeOperator type="BEFORE"/>
            <expressionSchema:timeExpression>
               <expressionSchema:timeVariable
               tvName="buyStock:buySingleStockRequest.buyBeforeDateTime"/>
            </expressionSchema:timeExpression>
         </expressionSchema:timeExpression>
      </expressionSchema:booleanExpression>
   </wsol:constraint>
   <wsol:constraint name="C50" xsi:type="qosSchema:QoS" service="buyStock:buyStockService"
   portOrPortType="buyStock:buyStockServicePort" operation="buyStock:buySingleStockOperation">
      <expressionSchema:booleanExpression>
         <expressionSchema:arithmeticWithUnitExpression>
            <expressionSchema:QoSmetric metricType="QoSMetricOntology:ResponseTime"
            metricUnit="QoSMeasOntology:millisecond" service="buyStock:buyStockService"
            portOrPortType="buyStock:buyStockServicePort"
            operation="buyStock:buySingleStockOperation" measuredBy="WSOL_INTERNAL"/>
         </expressionSchema:arithmeticWithUnitExpression>
         <expressionSchema:arithmeticWithUnitComparator type="&lt;"/>
         <expressionSchema:arithmeticWithUnitExpression>
            <expressionSchema:arithmeticWithUnitVariable
            aWUVName="tns:CGT2.responseTime"/>
         </expressionSchema:arithmeticWithUnitExpression>
      </expressionSchema:booleanExpression>
   </wsol:constraint>
</wsol:CGT>
```

Figure 44 Example of a constraint group template (CGT) specified using the improved WSOL constraint group template (CGT) grammar

In the above example, a new constraint group template named *“CGT2”* is defined for a particular service (*“buyStockService”*), port (*“buyStockServicePort”*), and operation (*“buySingleStockOperation”*). The value of the attribute *“satisfiedConstraints*” specified in the

CGT definition is “ONEORMORE” which means that one or more constraints specified in the CGT have to be satisfied.

7 WSOL Service Offerings

WSOL 1.0 enabled specification of the name of a service offering, the applicability domain of the service offering (i.e., the name of the Web Service for which the service offering is defined), the name of another service offering which the service offering extends, and the name of an accounting party for the service offering. WSOL 1.1 also enables specification of subscription information, validity period information, and auto manipulation information for a service offering. The need for these improvements was also mentioned in [5].

As stated earlier in this document, WSOL 1.1 enables specification of subscription information and the validity period information for a service offering using the general <*statement*> construct. Using this approach, it could be possible that multiple statements containing subscription information or validity period information for a single service offering are specified. So, this approach for specification of subscription information and the validity period information for a service offering does not seem to be efficient and WSOL 1.1 has now been modified to enable such specification in the definition of a service offering itself. Since some WSOL examples of specification of subscription information and the validity period information for a service offering have already been developed using the general <*statement*> construct, this method of specification will not be considered obsolete right now to enable backward compatibility. (However, it might become obsolete in future versions of WSOL.)

In order to enable specification of subscription information for a service offering, three optional attributes (*"subscriptionAmount"*, *"subscriptionCurrency"*, and *"subscriptionPeriod"*) have been added to the definition of the WSOL service offering. These three attributes enables specification of the price that has to be paid by a Web Service consumer for using a particular service offering for a specified duration of time. If no subscription amount is specified, this means that there is no subscription. If no subscription period is specified, it is assumed that subscription is valid until switching of service offerings (initiated by the consumer or the provider).

In order to enable specification of validity period information for a service offering, two optional attributes (*"duration"* and *"expiration"*) have been added to the definition of the WSOL service offering. These two attributes enables specification of the validity duration (i.e., the duration of time for which a service offering is valid) and the validity expiration date and time (i.e., the date and time when a service offering validity will expire) of a particular service offering. If no validity period information is specified for a service offering, then, it is to be assumed that the particular service offering is valid forever.

The auto manipulation information is used in algorithms for dynamic manipulation of service offerings. In order to enable specification of auto manipulation information for a service offering, an optional attribute (*"autoManipulation"*) has been added to the definition of the WSOL service offering. The value of this attribute can be either *"true"* or *"false"*. The default value for this attribute is *"false"*. If the value of the *"autoManipulation"* attribute specified for a service offering is *"true"*, then, it denotes that the Web Service provider first performs manipulation of the service offering and then notifies the Web Service consumer about the manipulation. On the other hand, if the value of the *"autoManipulation"* attribute specified for a

service offering is *"false"*, then, it denotes that the Web Service provider first asks for confirmation from the Web Service consumer and only then (after a positive confirmation) performs manipulation of the service offering. The former approach to dynamic manipulation (i.e., when the *"autoManipulation"* attribute is *"true"*) is faster, while in the latter approach (i.e., when the *"autoManipulation"* attribute is *"false"*) the consumer has more control.

The modified XML grammar for the specification of a WSOL service offering is shown in Figure 45.

```
<xsd:element name="serviceOffering" type="wsol:serviceOfferingType"/>
  <xsd:complexType name="serviceOfferingType">
    <xsd:sequence>
      <xsd:element ref="wsol:externalOperationCall" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:constraint" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:subscription" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:price" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:priceDefault" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:penalty" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:penaltyDefault" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:managementResponsibility" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:statement" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:include" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:CG" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:instantiate" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:NCName" use="required"/>
    <xsd:attribute name="extends" type="xsd:QName" default="WSOL-NONE"/>
    <xsd:attribute name="service" type="xsd:QName" use="required"/>
    <xsd:attribute name="accountingParty" type="xsd:anyURI" default="WSOL-SUPPLIERWS"/>
    <xsd:attribute name="duration" type="xsd:duration" use="optional"/>
    <xsd:attribute name="expiration" type="xsd:dateTime" use="optional"/>
    <xsd:attribute name="subscriptionAmount" type="xsd:double" use="optional"/>
    <xsd:attribute name="subscriptionCurrency" type="xsd:QName" use="optional"/>
    <xsd:attribute name="subscriptionPeriod" type="xsd:duration" use="optional"/>
    <xsd:attribute name="autoManipulation" type="xsd:boolean" use="optional" default="true"/>
  </xsd:complexType>
```

Figure 45 Modified XML grammar for the specification of a WSOL service offering

An example of a WSOL service offering specified using the modified XML grammar is given in Figure 46.

```
<wsol:serviceOffering name="SO5" service="buyStock:buyStockService" duration="P1Y"
expiration="2005-12-31T00:00:00-05:00" subscriptionAmount="50"
subscriptionCurrency="currencyOntology:Dollar" subscriptionPeriod="P1Y"
autoManipulation="true">
  <wsol:constraint name="QoScons5" xsi:type="qosSchema:QoS" service="buyStock:buyStockService"
  portOrPortType="buyStock:buyStockServicePort" operation="buyStock:buySingleStockOperation">
    <expressionSchema:booleanExpression>
      <expressionSchema:booleanExpression>
        <expressionSchema:booleanExpressionRef name="tns:SO1.QoScons1.BE1"/>
      </expressionSchema:booleanExpression>
      <expressionSchema:binaryBooleanOperator type="AND"/>
      <expressionSchema:booleanExpression>
        <expressionSchema:arithmeticWithUnitExpression>
          <expressionSchema:QoSmetric metricType="QoSMetricOntology:ResponseTime"
          metricUnit="QoSMeasOntology:millisecond" service="buyStock:buyStockService"
          portOrPortType="buyStock:buyStockServicePort"
          operation="buyStock:buySingleStockOperation" measuredBy="WSOL_INTERNAL"/>
        </expressionSchema:arithmeticWithUnitExpression>
        <expressionSchema:arithmeticWithUnitComparator type=">"/>
        <expressionSchema:arithmeticWithUnitExpression>
          <wsol:numberWithUnitConstant>
            <wsol:number value="5"/>
            <wsol:unit type="QoSMeasOntology:millisecond"/>
          </wsol:numberWithUnitConstant>
        </expressionSchema:arithmeticWithUnitExpression>
      </expressionSchema:booleanExpression>
    </expressionSchema:booleanExpression>
  </wsol:constraint>
</wsol:serviceOffering>
```

Figure 46 Example of a WSOL service offering specified using the modified XML grammar

In the above example, a service offering named *"SO5"* is defined using the modified XML grammar for the specification of a WSOL service offering. This service offering *"SO5"* definition contains subscription information, validity period information, and auto manipulation information for the service offering. The example service offering *"SO5"* is valid for duration of one year and it expires on 31st December 2005 at midnight. Also, a Web Service consumer has to pay a subscription amount of 50 Dollar to the Web Service provider for using this service offering for a period of one year. The value of the *"autoManipulation"* attribute of the example service offering *"SO5"* is *"true"* which means that the Web Service provider first performs manipulation of the service offering and then notifies the Web Service consumer about the manipulation.

8 Conclusion and Future Work

Research and development work has been done on the WSOL grammar and the "Premier" WSOL parser. Both the WSOL grammar and the WSOL parser were improved and extended. The WSOL grammar can now enable specification of periodic QoS constraints and the evaluation period for QoS constraints. Further, it can enable specification of dynamic relationships between service offerings that include complex expression, naming of all kinds of WSOL expressions, and referencing all kinds of WSOL expressions from other WSOL constraints. The <*serviceOffering*> element was extended with attributes for the specification of subscription information, validity period information, and auto manipulation information for WSOL service offerings. WSOL 1.1 also supports specification of existing types of WSOL statements and additional penalty statements using the general <*statement*> construct, and specification of CGs and CGTs in which only some constraints have to be satisfied. The WSOL parser can now validate specifications of future-conditions and service offerings dynamic relationships as well as some of the new WSOL specifications developed using the improved WSOL grammar. For example, it can validate specifications of periodic QoS constraints, QoS constraints with evaluation periods, dynamic relationships between service offerings that include complex expressions, and WSOL statements and additional penalty statements specified using the general <*statement*> construct, and can detect and report syntax and semantic errors in them. The improvements in the WSOL grammar and the WSOL parser were verified using the improved existing WSOL example and some new WSOL examples developed using the improved WSOL grammar.

The improvement of the "Premier" WSOL parser to enable validation of named WSOL expressions, referenced WSOL expressions from other WSOL constraints, specifications of CGs and CGTs in which only some constraints have to be satisfied, and specifications of service offerings for which subscription information, validity period information, and auto manipulation information has been specified has been left for future work due to time limitations. Further, WSOL parser also has to be extended with the implementation of checks for subdomains and specialization of domains. We plan to make WSOL compatible with WSDL version 1.2 in the future. We have also considered a number of other improvements of WSOL, but had to leave them for future versions of WSOL. Some examples of these future WSOL improvements are specification of service fees paid to management third parties, specification of periodic future-conditions, specification of real subcontracts (as discussed in [4]), improvement of the specification of external ontologies and use of WSOL expressions in these schemas. Some examples of future improvements of the "Premier" WSOL parser are implementation of partial instantiation of a CGT, generation of parsing descriptors, and improvement of the code readability and documentation. The development of a full WSOL compiler for the WSOL language has been left for future work. This compiler would use the already developed "Premier" WSOL parser and would add automatic generation of constraint-checking code for run-time monitoring, metering, and evaluation of WSOL constraints.

9 References

1. Patel, K. "XML Grammar and Parser for the Web Service Offerings Language", *Master of Applied Science thesis*, The Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada, January 2003. On-line at: http://www.sce.carleton.ca/netmanage/papers/KrutiPatelThesisFinal.pdf
2. Tosic, V., Pagurek, B., Patel, K. "WSOL – A Language for the Formal Specification of Various Constraints and Classes of Service for Web Services", *Technical Report OCIECE-02-06*, Ottawa-Carleton Institute for Electrical and Computer Engineering - OCIECE, November 2002.
3. Tosic, V., Pagurek, B., Patel, K. "WSOL – A Language for the Formal Specification of Classes of Service for Web Services". In *Proc. of ICWS'03 (The First International Conference on Web Services)*, Las Vegas, USA, June 23-26, 2003, CSREA Press, pp. 375-381, 2003. An early version of this paper was published as *Research Report SCE-03-05*, The Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada, February 2003.
4. Tosic, V., Patel, K., Pagurek, B. "Reusability Constructs in the Web Service Offerings Language (WSOL) [Second Extended Revision]". *Research Report SCE-03-21*, The Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada, September 2003. Shorter version of this paper: "Reusability Constructs in the Web Service Offerings Language (WSOL)" to be published in *Proc. of the Workshop on Web Services, e-Business, and the Semantic Web – WES* 2003 (at CAiSE'03), Bussler, C. et al. (eds.), Velden, Austria, June 2003, Springer-Verlag, Lecture Notes in Computer Science (LNCS). Early versions of this paper were published as *Research Report SCE-03-14* (May 2003) and *Research Report SCE-03-08* (April 2003).
5. Tosic, V., Pagurek, B., Patel, K., Esfandiari, B., Ma, W. "Management Applications of the Web Service Offerings Language (WSOL)". Submitted upon invitation to *Information Systems*, Elsevier. An early version of this paper was published in *Proc. of CAiSE'03 (The 15th International Conference on Advanced Information Systems Engineerin*g), Velden, Austria, June 16-20, 2003. Springer-Verlag, Lecture Notes in Computer Science (LNCS), No. 2681, pp. 468-484, 2003. Early versions of this paper were published as: Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada, *Research Report SCE-03-29* (October 2003), *SCE-03-09* (May 2003), and *SCE-03-04* (February 2003).
6. Tosic, V., Ma, W., Pagurek, B., Esfandiari, B. "On the Dynamic Manipulation of Classes of Service for XML Web Services". In *Proc. of the 10th Hewlett-Packard Open View University Association (HP-OVUA) Workshop*, Geneva, Switzerland, July 6-9, 2003. A version of this paper was published as *Research Report SCE-03-15*, The Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada, June 2003.
7. Tosic, V., Ma, W., Pagurek, B., Esfandiari, B. "Web Services Offerings Infrastructure (WSOI) - A Management Infrastructure for XML Web Services". Submitted for publication. Also published as *Research Report SCE-03-19*, The Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada, August 2003.

Appendix A: WSOL 1.1 Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
xmlns:wsol="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
<xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
schemaLocation="ExpressionSchema.xsd"/>
  <xsd:element name="WSOLdefinitions" type="wsol:WSOLdefinitionsType"/>
  <xsd:complexType name="WSOLdefinitionsType">
    <xsd:sequence>
      <xsd:element ref="wsol:import" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:externalOperationCall" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:constraint" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:subscription" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:price" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:priceDefault" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:penalty" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:penaltyDefault" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:managementResponsibility" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:statement" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:include" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:CG" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:CGT" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:instantiate" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:serviceOffering" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:any namespace="##other" processContents="strict" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="targetNamespace" type="xsd:anyURI" use="required"/>
  </xsd:complexType>
  <xsd:element name="import" type="wsol:importType"/>
  <xsd:complexType name="importType">
    <xsd:attribute name="namespace" type="xsd:anyURI" use="required"/>
    <xsd:attribute name="location" type="xsd:anyURI" use="required"/>
  </xsd:complexType>
  <xsd:element name="serviceOffering" type="wsol:serviceOfferingType"/>
  <xsd:complexType name="serviceOfferingType">
    <xsd:sequence>
      <xsd:element ref="wsol:externalOperationCall" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:constraint" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:subscription" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:price" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:priceDefault" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:penalty" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:penaltyDefault" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:managementResponsibility" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:statement" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:include" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:CG" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:instantiate" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:NCName" use="required"/>
```

```
    <xsd:attribute name="extends" type="xsd:QName" default="WSOL-NONE"/>
    <xsd:attribute name="service" type="xsd:QName" use="required"/>
    <xsd:attribute name="accountingParty" type="xsd:anyURI" default="WSOL-SUPPLIERWS"/>
    <xsd:attribute name="duration" type="xsd:duration" use="optional"/>
    <xsd:attribute name="expiration" type="xsd:dateTime" use="optional"/>
    <xsd:attribute name="subscriptionAmount" type="xsd:double" use="optional"/>
    <xsd:attribute name="subscriptionCurrency" type="xsd:QName" use="optional"/>
    <xsd:attribute name="subscriptionPeriod" type="xsd:duration" use="optional"/>
    <xsd:attribute name="autoManipulation" type="xsd:boolean" use="optional" default="true"/>
</xsd:complexType>
<xsd:element name="externalOperationCall" type="wsol:externalOperationCallType"/>
<xsd:complexType name="externalOperationCallType">
    <xsd:sequence>
        <xsd:element ref="wsol:callList" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="callID" type="xsd:NCName" use="required"/>
    <xsd:attribute name="calledOperDescription" type="xsd:QName" use="required"/>
    <xsd:attribute name="calledOperImplementation" type="xsd:QName" use="required"/>
    <xsd:attribute name="service" type="xsd:QName" use="required"/>
    <xsd:attribute name="portOrPortType" type="xsd:QName" use="required"/>
    <xsd:attribute name="operation" type="xsd:QName" use="required"/>
</xsd:complexType>
<xsd:element name="callList" type="wsol:callListType"/>
<xsd:complexType name="callListType">
    <xsd:sequence>
        <xsd:element ref="wsol:parameterValue" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:element name="parameterValue" type="wsol:parameterValueType"/>
<xsd:complexType name="parameterValueType">
    <xsd:sequence>
        <xsd:element ref="wsol:partName"/>
        <xsd:element ref="wsol:paramValue"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:element name="partName" type="wsol:partNameType"/>
<xsd:complexType name="partNameType">
    <xsd:attribute name="name" type="xsd:QName" use="required"/>
</xsd:complexType>
<xsd:element name="paramValue" type="wsol:paramValueType"/>
<xsd:complexType name="paramValueType">
    <xsd:choice>
        <xsd:element ref="expressionSchema:arithmeticExpression"/>
        <xsd:element ref="expressionSchema:stringExpression"/>
    </xsd:choice>
</xsd:complexType>
<xsd:element name="constraint" type="wsol:constraintType"/>
<xsd:complexType name="constraintType">
    <xsd:attribute name="name" type="xsd:NCName" use="required"/>
    <xsd:attribute name="service" type="xsd:QName" use="required"/>
    <xsd:attribute name="portOrPortType" type="xsd:QName" use="optional" default="WSOL-ALL"/>
    <xsd:attribute name="operation" type="xsd:QName" use="optional" default="WSOL-ALL"/>
</xsd:complexType>
<xsd:element name="statement" type="wsol:statementType"/>
<xsd:complexType name="statementType">
    <xsd:attribute name="name" type="xsd:NCName" use="required"/>
```

```
  </xsd:complexType>
  <xsd:element name="CG" type="wsol:CGType"/>
  <xsd:complexType name="CGType">
    <xsd:sequence>
      <xsd:element ref="wsol:externalOperationCall" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:constraint" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:subscription" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:price" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:priceDefault" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:penalty" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:penaltyDefault" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:managementResponsibility" minOccurs="0"/>
      <xsd:element ref="wsol:statement" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:include" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:CG" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:instantiate" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:NCName" use="required"/>
    <xsd:attribute name="extends" type="xsd:QName" use="optional" default="WSOL-NONE"/>
    <xsd:attribute name="service" type="xsd:QName" use="required"/>
    <xsd:attribute name="portOrPortType" type="xsd:QName" use="required"/>
    <xsd:attribute name="operation" type="xsd:QName" use="required"/>
<xsd:attribute name="satisfiedConstraints" type="wsol:satisfiedAttributeValues" use="optional" default="ALL"/>
  </xsd:complexType>
  <xsd:simpleType name="satisfiedAttributeValues">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="ALL"/>
      <xsd:enumeration value="ONEORMORE"/>
      <xsd:enumeration value="EXACTLYONE"/>
      <xsd:enumeration value="NONE"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:element name="instantiate" type="wsol:instantiateType"/>
  <xsd:complexType name="instantiateType">
    <xsd:sequence>
      <xsd:element ref="wsol:parmValue" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="CGTName" type="xsd:QName" use="required"/>
    <xsd:attribute name="resService" type="xsd:QName" use="required"/>
    <xsd:attribute name="resPortOrPortType" type="xsd:QName" use="required"/>
    <xsd:attribute name="resOperation" type="xsd:QName" use="required"/>
    <xsd:attribute name="resCGName" type="xsd:NCName" use="required"/>
  </xsd:complexType>
  <xsd:element name="parmValue" type="wsol:parmValueType"/>
  <xsd:complexType name="parmValueType">
    <xsd:sequence>
      <xsd:element ref="wsol:numberWithUnitConstant" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:QName" use="required"/>
    <xsd:attribute name="value" type="xsd:QName"/>
  </xsd:complexType>
  <xsd:element name="include" type="wsol:includeType"/>
  <xsd:complexType name="includeType">
    <xsd:attribute name="constructName" type="xsd:QName" use="required"/>
    <xsd:attribute name="resService" type="xsd:QName" use="required"/>
    <xsd:attribute name="resPortOrPortType" type="xsd:QName" use="required"/>
```

```
      <xsd:attribute name="resOperation" type="xsd:QName" use="required"/>
      <xsd:attribute name="resName" type="xsd:NCName" use="required"/>
    </xsd:complexType>
    <xsd:element name="CGT" type="wsol:CGTType"/>
    <xsd:complexType name="CGTType">
      <xsd:sequence>
        <xsd:element ref="wsol:parameter" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="wsol:externalOperationCall" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="wsol:constraint" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="wsol:subscription" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="wsol:price" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="wsol:priceDefault" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="wsol:penalty" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="wsol:penaltyDefault" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="wsol:managementResponsibility" minOccurs="0"/>
        <xsd:element ref="wsol:statement" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="wsol:include" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="wsol:CG" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="wsol:instantiate" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="name" type="xsd:NCName" use="required"/>
      <xsd:attribute name="extends" type="xsd:QName" default="WSOL-NONE"/>
      <xsd:attribute name="service" type="xsd:QName" use="required"/>
      <xsd:attribute name="portOrPortType" type="xsd:QName" use="required"/>
      <xsd:attribute name="operation" type="xsd:QName" use="required"/>
<xsd:attribute name="satisfiedConstraints" type="wsol:satisfiedAttributeValues" use="optional" default="ALL"/>
    </xsd:complexType>
    <xsd:element name="parameter" type="wsol:parameterType"/>
    <xsd:complexType name="parameterType">
      <xsd:attribute name="name" type="xsd:NCName" use="required"/>
      <xsd:attribute name="dataType" type="xsd:QName" use="required"/>
      <xsd:attribute name="unit" type="xsd:QName" default="WSOL-NOUNIT"/>
    </xsd:complexType>
    <xsd:element name="subscription" type="wsol:subscriptionType"/>
    <xsd:complexType name="subscriptionType">
      <xsd:sequence>
        <xsd:element ref="wsol:numberWithUnitConstant"/>
        <xsd:element ref="wsol:subscriptionDuration"/>
      </xsd:sequence>
      <xsd:attribute name="name" type="xsd:NCName" use="required"/>
    </xsd:complexType>
    <xsd:element name="priceDefault" type="wsol:priceDefaultType"/>
    <xsd:complexType name="priceDefaultType">
      <xsd:sequence>
        <xsd:element ref="wsol:numberWithUnitConstant"/>
      </xsd:sequence>
      <xsd:attribute name="name" type="xsd:NCName" use="required"/>
    </xsd:complexType>
    <xsd:element name="penaltyDefault" type="wsol:penaltyDefaultType"/>
    <xsd:complexType name="penaltyDefaultType">
      <xsd:sequence>
        <xsd:element ref="wsol:numberWithUnitConstant"/>
      </xsd:sequence>
      <xsd:attribute name="name" type="xsd:NCName" use="required"/>
    </xsd:complexType>
    <xsd:element name="price" type="wsol:priceType"/>
```

```
<xsd:complexType name="priceType">
   <xsd:sequence>
      <xsd:element ref="wsol:numberWithUnitConstant"/>
   </xsd:sequence>
   <xsd:attribute name="name" type="xsd:NCName" use="required"/>
   <xsd:attribute name="service" type="xsd:QName" use="required"/>
   <xsd:attribute name="portOrPortType" type="xsd:QName" use="required"/>
   <xsd:attribute name="operation" type="xsd:QName" use="required"/>
</xsd:complexType>
<xsd:element name="penalty" type="wsol:penaltyType"/>
<xsd:complexType name="penaltyType">
   <xsd:sequence>
      <xsd:element ref="wsol:numberWithUnitConstant"/>
   </xsd:sequence>
   <xsd:attribute name="name" type="xsd:NCName" use="required"/>
   <xsd:attribute name="service" type="xsd:QName" use="required"/>
   <xsd:attribute name="portOrPortType" type="xsd:QName" use="required"/>
   <xsd:attribute name="operation" type="xsd:QName" use="required"/>
</xsd:complexType>
<xsd:element name="subscriptionDuration" type="wsol:subscriptionDurationType"/>
<xsd:complexType name="subscriptionDurationType">
   <xsd:attribute name="duration" type="xsd:duration" use="required"/>
</xsd:complexType>
<xsd:complexType name="numberWithUnit">
   <xsd:sequence>
      <xsd:element ref="wsol:number"/>
      <xsd:element ref="wsol:unit"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:element name="unit" type="wsol:unitType"/>
<xsd:complexType name="unitType">
   <xsd:attribute name="type" type="xsd:QName" use="required"/>
</xsd:complexType>
<xsd:element name="number" type="wsol:numberType"/>
<xsd:complexType name="numberType">
   <xsd:attribute name="value" type="xsd:double" use="required"/>
</xsd:complexType>
<xsd:element name="numberWithUnitConstant" type="wsol:numberWithUnit"/>
<xsd:element name="managementResponsibility" type="wsol:managementResponsibilityType"/>
<xsd:complexType name="managementResponsibilityType">
   <xsd:sequence>
      <xsd:element ref="wsol:supplierResponsibility" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:consumerResponsibility" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="wsol:independentResponsibility" minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="name" type="xsd:NCName" use="required"/>
</xsd:complexType>
<xsd:element name="supplierResponsibility" type="wsol:supplierResponsibilityType"/>
<xsd:complexType name="supplierResponsibilityType">
   <xsd:attribute name="scope" type="xsd:QName" use="required"/>
</xsd:complexType>
<xsd:element name="consumerResponsibility" type="wsol:consumerResponsibilityType"/>
<xsd:complexType name="consumerResponsibilityType">
   <xsd:attribute name="scope" type="xsd:QName" use="required"/>
</xsd:complexType>
<xsd:element name="independentResponsibility" type="wsol:independentResponsibilityType"/>
```

```
   <xsd:complexType name="independentResponsibilityType">
      <xsd:attribute name="scope" type="xsd:QName" use="required"/>
      <xsd:attribute name="entity" type="xsd:anyURI" use="required"/>
   </xsd:complexType>
</xsd:schema>
```

Appendix B: Expression Schema for WSOL 1.1

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:wsol="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
xmlns:sodr="http://www.sce.carleton.ca/~kpatel/WSOL/SODRSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
<xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
schemaLocation="WSOLSchema.xsd"/>
<xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/SODRSchema"
schemaLocation="D:/users/kpatel/krutiMay-Aug2003/FinalXMLSchemasAndExamples/SODRSchema.xsd"/>
  <xsd:element name="booleanExpression" type="expressionSchema:booleanExpressionType"/>
  <xsd:complexType name="booleanExpressionType">
    <xsd:choice>
      <xsd:element ref="expressionSchema:booleanConstant"/>
      <xsd:element ref="expressionSchema:booleanVariable"/>
      <xsd:element ref="expressionSchema:externalOperationResult"/>
      <xsd:element ref="expressionSchema:booleanExpression"/>
      <xsd:element ref="expressionSchema:quantifiedExpression"/>
      <xsd:element ref="expressionSchema:booleanExpressionRef"/>
      <xsd:element ref="sodr:unsatisfiedConstraintRef"/>
      <xsd:sequence>
        <xsd:element ref="expressionSchema:unaryBooleanOperator"/>
        <xsd:element ref="expressionSchema:booleanExpression"/>
      </xsd:sequence>
      <xsd:sequence>
        <xsd:element ref="expressionSchema:booleanExpression"/>
        <xsd:element ref="expressionSchema:binaryBooleanOperator"/>
        <xsd:element ref="expressionSchema:booleanExpression"/>
      </xsd:sequence>
      <xsd:sequence>
        <xsd:element ref="expressionSchema:booleanExpression"/>
        <xsd:element ref="expressionSchema:booleanComparator"/>
        <xsd:element ref="expressionSchema:booleanExpression"/>
      </xsd:sequence>
      <xsd:sequence>
        <xsd:element ref="expressionSchema:arithmeticExpression"/>
        <xsd:element ref="expressionSchema:arithmeticComparator"/>
        <xsd:element ref="expressionSchema:arithmeticExpression"/>
      </xsd:sequence>
      <xsd:sequence>
        <xsd:element ref="expressionSchema:arithmeticWithUnitExpression"/>
        <xsd:element ref="expressionSchema:arithmeticWithUnitComparator"/>
        <xsd:element ref="expressionSchema:arithmeticWithUnitExpression"/>
      </xsd:sequence>
      <xsd:sequence>
        <xsd:element ref="expressionSchema:stringExpression"/>
        <xsd:element ref="expressionSchema:stringComparator"/>
        <xsd:element ref="expressionSchema:stringExpression"/>
      </xsd:sequence>
      <xsd:sequence>
        <xsd:element ref="expressionSchema:timeExpression"/>
```

```
        <xsd:element ref="expressionSchema:timeComparator"/>
        <xsd:element ref="expressionSchema:timeExpression"/>
      </xsd:sequence>
    </xsd:choice>
    <xsd:attribute name="name" type="xsd:NCName" use="optional"/>
  </xsd:complexType>
  <xsd:element name="booleanConstant" type="expressionSchema:booleanConstantType"/>
  <xsd:complexType name="booleanConstantType">
    <xsd:attribute name="type" type="expressionSchema:booleanConstantTypeChoice" use="required"/>
  </xsd:complexType>
  <xsd:simpleType name="booleanConstantTypeChoice">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="true"/>
      <xsd:enumeration value="false"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="unaryBooleanOperatorTypeChoice">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="NOT"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="binaryBooleanOperatorTypeChoice">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="AND"/>
      <xsd:enumeration value="OR"/>
      <xsd:enumeration value="XOR"/>
      <xsd:enumeration value="EQUIVALENT"/>
      <xsd:enumeration value="IMPLIES"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="booleanComparatorTypeChoice">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="=="/>
      <xsd:enumeration value="!="/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="quantifiedExpressionTypeChoice">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="FOR_ALL_IN"/>
      <xsd:enumeration value="EXISTS_IN"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="unaryArithmeticOperatorTypeChoice">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="-"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="binaryArithmeticOperatorTypeChoice">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="+"/>
      <xsd:enumeration value="-"/>
      <xsd:enumeration value="*"/>
      <xsd:enumeration value="/"/>
      <xsd:enumeration value="**"/>
    </xsd:restriction>
  </xsd:simpleType>
```

```
<xsd:simpleType name="arithmeticComparatorTypeChoice">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="&lt;"/>
    <xsd:enumeration value=">"/>
    <xsd:enumeration value="=="/>
    <xsd:enumeration value="&lt;="/>
    <xsd:enumeration value=">="/>
    <xsd:enumeration value="!="/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="arithmeticWithUnitComparatorTypeChoice">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="&lt;"/>
    <xsd:enumeration value=">"/>
    <xsd:enumeration value="=="/>
    <xsd:enumeration value="&lt;="/>
    <xsd:enumeration value=">="/>
    <xsd:enumeration value="!="/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="stringComparatorTypeChoice">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="&lt;"/>
    <xsd:enumeration value=">"/>
    <xsd:enumeration value="=="/>
    <xsd:enumeration value="&lt;="/>
    <xsd:enumeration value=">="/>
    <xsd:enumeration value="!="/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="timeOperatorTypeChoice">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="BEFORE"/>
    <xsd:enumeration value="AFTER"/>
    <xsd:enumeration value="DURING"/>
    <xsd:enumeration value="BETWEEN"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="timeComparatorTypeChoice">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="&lt;"/>
    <xsd:enumeration value=">"/>
    <xsd:enumeration value="=="/>
    <xsd:enumeration value="&lt;="/>
    <xsd:enumeration value=">="/>
    <xsd:enumeration value="!="/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="binaryArithmeticWithUnitOperator1TypeChoice">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="+"/>
    <xsd:enumeration value="-"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="binaryArithmeticWithUnitOperator2TypeChoice">
  <xsd:restriction base="xsd:string">
```

```
      <xsd:enumeration value="*"/>
      <xsd:enumeration value="/"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:element name="externalOperationResult" type="expressionSchema:externalOperationResultType"/>
  <xsd:complexType name="externalOperationResultType">
    <xsd:attribute name="callID" type="xsd:QName" use="required"/>
    <xsd:attribute name="resultPartName" type="xsd:QName" use="required"/>
  </xsd:complexType>
  <xsd:element name="unaryBooleanOperator" type="expressionSchema:unaryBooleanOperatorType"/>
  <xsd:complexType name="unaryBooleanOperatorType">
    <xsd:attribute name="type" type="expressionSchema:unaryBooleanOperatorTypeChoice" use="required"/>
  </xsd:complexType>
  <xsd:element name="binaryBooleanOperator" type="expressionSchema:binaryBooleanOperatorType"/>
  <xsd:complexType name="binaryBooleanOperatorType">
    <xsd:attribute name="type" type="expressionSchema:binaryBooleanOperatorTypeChoice" use="required"/>
  </xsd:complexType>
  <xsd:element name="booleanComparator" type="expressionSchema:booleanComparatorType"/>
  <xsd:complexType name="booleanComparatorType">
    <xsd:attribute name="type" type="expressionSchema:booleanComparatorTypeChoice" use="required"/>
  </xsd:complexType>
  <xsd:element name="quantifiedExpression" type="expressionSchema:quantifiedExpressionType"/>
  <xsd:complexType name="quantifiedExpressionType">
    <xsd:choice>
      <xsd:element ref="expressionSchema:booleanExpression"/>
      <xsd:element ref="expressionSchema:quantifiedExpressionRef"/>
    </xsd:choice>
    <xsd:attribute name="name" type="xsd:NCName" use="optional"/>
    <xsd:attribute name="type" type="expressionSchema:quantifiedExpressionTypeChoice" use="required"/>
    <xsd:attribute name="arrayVariableName" type="xsd:QName" use="required"/>
  </xsd:complexType>
  <xsd:element name="arithmeticExpression" type="expressionSchema:arithmeticExpressionType"/>
  <xsd:complexType name="arithmeticExpressionType">
    <xsd:choice>
      <xsd:element ref="expressionSchema:arithmeticConstant"/>
      <xsd:element ref="expressionSchema:arithmeticVariable"/>
      <xsd:element ref="expressionSchema:externalOperationResult"/>
      <xsd:element ref="expressionSchema:arithmeticExpression"/>
      <xsd:element ref="expressionSchema:arithmeticExpressionRef"/>
      <xsd:sequence>
        <xsd:element ref="expressionSchema:unaryArithmeticOperator"/>
        <xsd:element ref="expressionSchema:arithmeticExpression"/>
      </xsd:sequence>
      <xsd:sequence>
        <xsd:element ref="expressionSchema:arithmeticExpression"/>
        <xsd:element ref="expressionSchema:binaryArithmeticOperator"/>
        <xsd:element ref="expressionSchema:arithmeticExpression"/>
      </xsd:sequence>
    </xsd:choice>
    <xsd:attribute name="name" type="xsd:NCName" use="optional"/>
  </xsd:complexType>
  <xsd:element name="arithmeticConstant" type="expressionSchema:arithmeticConstantType"/>
  <xsd:complexType name="arithmeticConstantType">
    <xsd:choice>
      <xsd:element ref="expressionSchema:integerConstant"/>
      <xsd:element ref="expressionSchema:floatConstant"/>
```

```
        <xsd:element ref="expressionSchema:doubleConstant"/>
        <xsd:element ref="expressionSchema:longConstant"/>
      </xsd:choice>
    </xsd:complexType>
    <xsd:element name="QoSmetric" type="expressionSchema:QoSmetricType"/>
    <xsd:complexType name="QoSmetricType">
      <xsd:sequence>
        <xsd:element ref="expressionSchema:usedQoSmetric" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="metricType" type="xsd:QName" use="required"/>
      <xsd:attribute name="metricUnit" type="xsd:QName" use="required"/>
      <xsd:attribute name="service" type="xsd:QName" use="required"/>
      <xsd:attribute name="portOrPortType" type="xsd:QName" use="required"/>
      <xsd:attribute name="operation" type="xsd:QName" use="required"/>
      <xsd:attribute name="measuredBy" type="xsd:QName" use="required"/>
    </xsd:complexType>
    <xsd:element name="usedQoSmetric" type="expressionSchema:usedQoSmetricType"/>
    <xsd:complexType name="usedQoSmetricType">
      <xsd:attribute name="metricType" type="xsd:QName" use="required"/>
      <xsd:attribute name="metricUnit" type="xsd:QName" use="required"/>
      <xsd:attribute name="service" type="xsd:QName" use="required"/>
      <xsd:attribute name="portOrPortType" type="xsd:QName" use="required"/>
      <xsd:attribute name="operation" type="xsd:QName" use="required"/>
      <xsd:attribute name="measuredBy" type="xsd:QName" use="required"/>
    </xsd:complexType>
  <xsd:element name="arithmeticWithUnitExpression"
  type="expressionSchema:arithmeticWithUnitExpressionType"/>
    <xsd:complexType name="arithmeticWithUnitExpressionType">
      <xsd:choice>
        <xsd:element ref="expressionSchema:arithmeticWithUnitVariable"/>
        <xsd:element ref="expressionSchema:QoSmetric"/>
        <xsd:element ref="expressionSchema:arithmeticWithUnitExpressionRef"/>
        <xsd:sequence>
          <xsd:element ref="expressionSchema:arithmeticWithUnitExpression"/>
          <xsd:element ref="expressionSchema:binaryArithmeticWithUnitOperator1"/>
          <xsd:element ref="expressionSchema:arithmeticWithUnitExpression"/>
        </xsd:sequence>
        <xsd:sequence>
          <xsd:element ref="expressionSchema:arithmeticWithUnitExpression"/>
          <xsd:element ref="expressionSchema:binaryArithmeticWithUnitOperator2"/>
          <xsd:element ref="expressionSchema:arithmeticConstant"/>
        </xsd:sequence>
        <xsd:element ref="wsol:numberWithUnitConstant"/>
      </xsd:choice>
      <xsd:attribute name="name" type="xsd:NCName" use="optional"/>
    </xsd:complexType>
    <xsd:element name="unaryArithmeticOperator" type="expressionSchema:unaryArithmeticOperatorType"/>
    <xsd:complexType name="unaryArithmeticOperatorType">
      <xsd:attribute name="type" type="expressionSchema:unaryArithmeticOperatorTypeChoice" use="required"/>
    </xsd:complexType>
    <xsd:element name="binaryArithmeticOperator" type="expressionSchema:binaryArithmeticOperatorType"/>
    <xsd:complexType name="binaryArithmeticOperatorType">
      <xsd:attribute name="type" type="expressionSchema:binaryArithmeticOperatorTypeChoice" use="required"/>
    </xsd:complexType>
  <xsd:element name="binaryArithmeticWithUnitOperator1"
  type="expressionSchema:binaryArithmeticWithUnitOperator1Type"/>
```

```
   <xsd:complexType name="binaryArithmeticWithUnitOperator1Type">
<xsd:attribute name="type" type="expressionSchema:binaryArithmeticWithUnitOperator1TypeChoice"
use="required"/>
   </xsd:complexType>
<xsd:element name="binaryArithmeticWithUnitOperator2"
type="expressionSchema:binaryArithmeticWithUnitOperator2Type"/>
   <xsd:complexType name="binaryArithmeticWithUnitOperator2Type">
<xsd:attribute name="type" type="expressionSchema:binaryArithmeticWithUnitOperator2TypeChoice"
use="required"/>
   </xsd:complexType>
   <xsd:element name="arithmeticComparator" type="expressionSchema:arithmeticComparatorType"/>
   <xsd:complexType name="arithmeticComparatorType">
      <xsd:attribute name="type" type="expressionSchema:arithmeticComparatorTypeChoice" use="required"/>
   </xsd:complexType>
<xsd:element name="arithmeticWithUnitComparator"
type="expressionSchema:arithmeticWithUnitComparatorType"/>
   <xsd:complexType name="arithmeticWithUnitComparatorType">
<xsd:attribute name="type" type="expressionSchema:arithmeticWithUnitComparatorTypeChoice" use="required"/>
   </xsd:complexType>
   <xsd:element name="stringExpression" type="expressionSchema:stringExpressionType"/>
   <xsd:complexType name="stringExpressionType">
      <xsd:choice>
         <xsd:element ref="expressionSchema:stringConstant"/>
         <xsd:element ref="expressionSchema:stringVariable"/>
         <xsd:element ref="expressionSchema:externalOperationResult"/>
         <xsd:element ref="expressionSchema:stringExpression"/>
         <xsd:element ref="expressionSchema:stringExpressionRef"/>
      </xsd:choice>
      <xsd:attribute name="name" type="xsd:NCName" use="optional"/>
   </xsd:complexType>
   <xsd:element name="stringComparator" type="expressionSchema:stringComparatorType"/>
   <xsd:complexType name="stringComparatorType">
      <xsd:attribute name="type" type="expressionSchema:stringComparatorTypeChoice" use="required"/>
   </xsd:complexType>
   <xsd:element name="timeExpression" type="expressionSchema:timeExpressionType"/>
   <xsd:complexType name="timeExpressionType">
      <xsd:choice>
         <xsd:element ref="expressionSchema:timeConstant"/>
         <xsd:element ref="expressionSchema:timeVariable"/>
         <xsd:element ref="expressionSchema:externalOperationResult"/>
         <xsd:element ref="expressionSchema:timeExpression"/>
         <xsd:element ref="expressionSchema:timeExpressionRef"/>
         <xsd:sequence>
            <xsd:element ref="expressionSchema:timeOperator"/>
            <xsd:element ref="expressionSchema:timeExpression"/>
         </xsd:sequence>
      </xsd:choice>
      <xsd:attribute name="name" type="xsd:NCName" use="optional"/>
   </xsd:complexType>
   <xsd:element name="timeConstant" type="expressionSchema:timeConstantType"/>
   <xsd:complexType name="timeConstantType">
      <xsd:choice>
         <xsd:element ref="expressionSchema:time"/>
         <xsd:element ref="expressionSchema:date"/>
         <xsd:element ref="expressionSchema:date_and_time"/>
         <xsd:element ref="expressionSchema:time_duration"/>
```

```
      <xsd:element ref="expressionSchema:gregorianYear"/>
      <xsd:element ref="expressionSchema:gregorianYearMonth"/>
      <xsd:element ref="expressionSchema:gregorianMonthDay"/>
      <xsd:element ref="expressionSchema:gregorianMonth"/>
      <xsd:element ref="expressionSchema:gregorianDay"/>
    </xsd:choice>
  </xsd:complexType>
  <xsd:element name="timeOperator" type="expressionSchema:timeOperatorType"/>
  <xsd:complexType name="timeOperatorType">
    <xsd:attribute name="type" type="expressionSchema:timeOperatorTypeChoice" use="required"/>
  </xsd:complexType>
  <xsd:element name="timeComparator" type="expressionSchema:timeComparatorType"/>
  <xsd:complexType name="timeComparatorType">
    <xsd:attribute name="type" type="expressionSchema:timeComparatorTypeChoice" use="required"/>
  </xsd:complexType>
  <xsd:element name="integerConstant" type="expressionSchema:integerConstantType"/>
  <xsd:complexType name="integerConstantType">
    <xsd:attribute name="value" type="xsd:int" use="required"/>
  </xsd:complexType>
  <xsd:element name="floatConstant" type="expressionSchema:floatConstantType"/>
  <xsd:complexType name="floatConstantType">
    <xsd:attribute name="value" type="xsd:float" use="required"/>
  </xsd:complexType>
  <xsd:element name="doubleConstant" type="expressionSchema:doubleConstantType"/>
  <xsd:complexType name="doubleConstantType">
    <xsd:attribute name="value" type="xsd:double" use="required"/>
  </xsd:complexType>
  <xsd:element name="longConstant" type="expressionSchema:longConstantType"/>
  <xsd:complexType name="longConstantType">
    <xsd:attribute name="value" type="xsd:long" use="required"/>
  </xsd:complexType>
  <xsd:element name="time" type="expressionSchema:timeType"/>
  <xsd:complexType name="timeType">
    <xsd:attribute name="value" type="xsd:time" use="required"/>
  </xsd:complexType>
  <xsd:element name="date" type="expressionSchema:dateType"/>
  <xsd:complexType name="dateType">
    <xsd:attribute name="value" type="xsd:date" use="required"/>
  </xsd:complexType>
  <xsd:element name="date_and_time" type="expressionSchema:date_and_timeType"/>
  <xsd:complexType name="date_and_timeType">
    <xsd:attribute name="value" type="xsd:dateTime" use="required"/>
  </xsd:complexType>
  <xsd:element name="time_duration" type="expressionSchema:time_durationType"/>
  <xsd:complexType name="time_durationType">
    <xsd:attribute name="value" type="xsd:duration" use="required"/>
  </xsd:complexType>
  <xsd:element name="gregorianYear" type="expressionSchema:gregorianYearType"/>
  <xsd:complexType name="gregorianYearType">
    <xsd:attribute name="value" type="xsd:gYear" use="required"/>
  </xsd:complexType>
  <xsd:element name="gregorianYearMonth" type="expressionSchema:gregorianYearMonthType"/>
  <xsd:complexType name="gregorianYearMonthType">
    <xsd:attribute name="value" type="xsd:gYearMonth" use="required"/>
  </xsd:complexType>
  <xsd:element name="gregorianMonthDay" type="expressionSchema:gregorianMonthDayType"/>
```

```
    <xsd:complexType name="gregorianMonthDayType">
      <xsd:attribute name="value" type="xsd:gMonthDay" use="required"/>
    </xsd:complexType>
    <xsd:element name="gregorianMonth" type="expressionSchema:gregorianMonthType"/>
    <xsd:complexType name="gregorianMonthType">
      <xsd:attribute name="value" type="xsd:gMonth" use="required"/>
    </xsd:complexType>
    <xsd:element name="gregorianDay" type="expressionSchema:gregorianDayType"/>
    <xsd:complexType name="gregorianDayType">
      <xsd:attribute name="value" type="xsd:gDay" use="required"/>
    </xsd:complexType>
    <xsd:element name="booleanVariable" type="expressionSchema:booleanVariableType"/>
    <xsd:complexType name="booleanVariableType">
      <xsd:attribute name="bvName" type="xsd:QName" use="required"/>
    </xsd:complexType>
    <xsd:element name="arithmeticVariable" type="expressionSchema:arithmeticVariableType"/>
    <xsd:complexType name="arithmeticVariableType">
      <xsd:attribute name="avName" type="xsd:QName" use="required"/>
    </xsd:complexType>
    <xsd:element name="arithmeticWithUnitVariable" type="expressionSchema:arithmeticWithUnitVariableType"/>
    <xsd:complexType name="arithmeticWithUnitVariableType">
      <xsd:attribute name="aWUVName" type="xsd:QName" use="required"/>
    </xsd:complexType>
    <xsd:element name="stringVariable" type="expressionSchema:stringVariableType"/>
    <xsd:complexType name="stringVariableType">
      <xsd:attribute name="svName" type="xsd:QName" use="required"/>
    </xsd:complexType>
    <xsd:element name="timeVariable" type="expressionSchema:timeVariableType"/>
    <xsd:complexType name="timeVariableType">
      <xsd:attribute name="tvName" type="xsd:QName" use="required"/>
    </xsd:complexType>
    <xsd:element name="arrayVariable" type="expressionSchema:arrayVariableType"/>
    <xsd:complexType name="arrayVariableType">
      <xsd:attribute name="arrayVName" type="xsd:QName" use="required"/>
    </xsd:complexType>
    <xsd:element name="stringConstant" type="expressionSchema:stringConstantType"/>
    <xsd:complexType name="stringConstantType">
      <xsd:attribute name="value" type="xsd:string" use="required"/>
    </xsd:complexType>
    <xsd:element name="booleanExpressionRef" type="expressionSchema:booleanExpressionRefType"/>
    <xsd:complexType name="booleanExpressionRefType">
      <xsd:attribute name="name" type="xsd:QName" use="required"/>
    </xsd:complexType>
    <xsd:element name="arithmeticExpressionRef" type="expressionSchema:arithmeticExpressionRefType"/>
    <xsd:complexType name="arithmeticExpressionRefType">
      <xsd:attribute name="name" type="xsd:QName" use="required"/>
    </xsd:complexType>
  <xsd:element name="arithmeticWithUnitExpressionRef"
  type="expressionSchema:arithmeticWithUnitExpressionRefType"/>
    <xsd:complexType name="arithmeticWithUnitExpressionRefType">
      <xsd:attribute name="name" type="xsd:QName" use="required"/>
    </xsd:complexType>
    <xsd:element name="quantifiedExpressionRef" type="expressionSchema:quantifiedExpressionRefType"/>
    <xsd:complexType name="quantifiedExpressionRefType">
      <xsd:attribute name="name" type="xsd:QName" use="required"/>
    </xsd:complexType>
```

```
    <xsd:element name="stringExpressionRef" type="expressionSchema:stringExpressionRefType"/>
    <xsd:complexType name="stringExpressionRefType">
      <xsd:attribute name="name" type="xsd:QName" use="required"/>
    </xsd:complexType>
    <xsd:element name="timeExpressionRef" type="expressionSchema:timeExpressionRefType"/>
    <xsd:complexType name="timeExpressionRefType">
      <xsd:attribute name="name" type="xsd:QName" use="required"/>
    </xsd:complexType>
</xsd:schema>
```

Appendix C: Constraint Schemas Defined for WSOL 1.1

C.1 Pre-condition Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Generated by Turbo XML 2.3.1.100. Conforms to w3c http://www.w3.org/2001/XMLSchema-->
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/PreConditionSchema"
xmlns:preConditionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/PreConditionSchema"
xmlns:wsol="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
schemaLocation="Schemas/WSOLSchema.xsd"/>
  <import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/ExpressionSchema"
schemaLocation="Schemas/ExpressionSchema.xsd"/>
  <complexType name="preCondition">
    <complexContent>
      <extension base="wsol:constraintType">
        <all>
          <element ref="expressionSchema:booleanExpression"/>
        </all>
      </extension>
    </complexContent>
  </complexType>
</schema>
```

C.2 Post-condition Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/PostConditionSchema"
xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/PostConditionSchema"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:wsol="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
xmlns:postConditionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/PostConditionSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/ExpressionSchema"
schemaLocation="Schemas/ExpressionSchema.xsd"/>
  <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
schemaLocation="Schemas/WSOLSchema.xsd"/>
  <xsd:complexType name="postCondition">
    <xsd:complexContent>
      <xsd:extension base="wsol:constraintType">
        <xsd:all>
          <xsd:element ref="expressionSchema:booleanExpression"/>
        </xsd:all>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```

C.3 Future-condition Schema

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<xsd:schema targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/FutureConditionSchema"
xmlns:futureConditionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/FutureConditionSchema"
xmlns:wsol="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/FutureConditionSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
schemaLocation="Schemas/ExpressionSchema.xsd"/>
  <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
schemaLocation="Schemas/WSOLSchema.xsd"/>
  <xsd:complexType name="futureCondition">
    <xsd:complexContent>
      <xsd:extension base="wsol:constraintType">
        <xsd:sequence>
          <xsd:choice>
            <xsd:element ref="evaluateAt"/>
            <xsd:element ref="evaluateAfter"/>
          </xsd:choice>
          <xsd:element ref="expressionSchema:booleanExpression"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="evaluateAt">
    <xsd:complexType>
      <xsd:attribute name="dateTime" type="xsd:dateTime" use="required"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="evaluateAfter">
    <xsd:complexType>
      <xsd:attribute name="duration" type="xsd:duration" use="required"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

C.4 Invariant Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/InvariantSchema"
xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/InvariantSchema"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:wsol="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
xmlns:invariantSchema="http://www.sce.carleton.ca/~kpatel/WSOL/InvariantSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/ExpressionSchema"
schemaLocation="Schemas/ExpressionSchema.xsd"/>
  <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
schemaLocation="Schemas/WSOLSchema.xsd"/>
  <xsd:complexType name="invariant">
    <xsd:complexContent>
      <xsd:extension base="wsol:constraintType">
        <xsd:all>
          <xsd:element ref="expressionSchema:booleanExpression"/>
        </xsd:all>
      </xsd:extension>
```

```
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```

C.5 QoS Constraint Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/QoS_Schema"
xmlns:qosSchema="http://www.sce.carleton.ca/~kpatel/WSOL/QoS_Schema"
xmlns:wsol="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/QoS_Schema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
schemaLocation="Schemas/ExpressionSchema.xsd"/>
  <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
schemaLocation="Schemas/WSOLSchema.xsd"/>
  <xsd:complexType name="QoS">
    <xsd:complexContent>
      <xsd:extension base="wsol:constraintType">
        <xsd:sequence>
          <xsd:element ref="expressionSchema:booleanExpression"/>
        </xsd:sequence>
        <xsd:attribute name="evalPeriod" type="xsd:positiveInteger" use="optional" default="1"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```

C.6 Periodic QoS Constraint Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/PeriodicQoSSchema"
xmlns:periodicQoSSchema="http://www.sce.carleton.ca/~kpatel/WSOL/PeriodicQoSSchema"
xmlns:wsol="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/PeriodicQoSSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
schemaLocation="Schemas/ExpressionSchema.xsd"/>
  <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
schemaLocation="Schemas/WSOLSchema.xsd"/>
  <xsd:complexType name="periodicQoS">
    <xsd:complexContent>
      <xsd:extension base="wsol:constraintType">
        <xsd:sequence>
          <xsd:element ref="evaluationStartTime"/>
          <xsd:element ref="evaluationPeriod"/>
          <xsd:element ref="evaluationStopTime"/>
          <xsd:element ref="maxRepetitions"/>
          <xsd:element ref="expressionSchema:booleanExpression"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
```

```
  <xsd:element name="evaluationStartTime">
    <xsd:complexType>
      <xsd:attribute name="time" type="xsd:dateTime" use="required"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="evaluationPeriod">
    <xsd:complexType>
      <xsd:attribute name="interval" type="xsd:duration" use="required"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="evaluationStopTime">
    <xsd:complexType>
      <xsd:attribute name="time" type="xsd:dateTime" use="required"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="maxRepetitions">
    <xsd:complexType>
      <xsd:attribute name="number" type="xsd:int" use="required"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

C.7 Access Right Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/AccessRightSchema"
xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/AccessRightSchema"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:wsol="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
xmlns:accessRightSchema="http://www.sce.carleton.ca/~kpatel/WSOL/AccessRightSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/ExpressionSchema"
schemaLocation="Schemas/ExpressionSchema.xsd"/>
  <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
schemaLocation="Schemas/WSOLSchema.xsd"/>
  <xsd:complexType name="accessRight">
    <xsd:complexContent>
      <xsd:extension base="wsol:constraintType">
        <xsd:all>
          <xsd:element ref="expressionSchema:booleanExpression"/>
        </xsd:all>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```

Appendix D: Statement Schemas Defined for WSOL 1.1

D.1 Service Offering Validity Statement Schema

```
<?xml version="1.0"?>
<xsd:schema targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/SOValiditySchema"
xmlns:soValiditySchema="http://www.sce.carleton.ca/~kpatel/WSOL/SOValiditySchema"
xmlns:wsol="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/SOValiditySchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
schemaLocation="Schemas/ExpressionSchema.xsd"/>
   <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
schemaLocation="Schemas/WSOLSchema.xsd"/>
   <xsd:complexType name="soValidity">
      <xsd:complexContent>
         <xsd:extension base="wsol:statementType">
            <xsd:sequence>
               <xsd:element ref="soName" maxOccurs="unbounded"/>
               <xsd:element ref="soValidityDuration"/>
               <xsd:element ref="soValidityExpiration"/>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
   <xsd:element name="soName">
      <xsd:complexType>
         <xsd:attribute name="name" type="xsd:QName" use="required"/>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="soValidityDuration">
      <xsd:complexType>
         <xsd:attribute name="duration" type="xsd:duration" use="required"/>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="soValidityExpiration">
      <xsd:complexType>
         <xsd:attribute name="expDateTime" type="xsd:dateTime" use="required"/>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

D.2 Subscription Statement Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/SubscriptionSchema"
xmlns:subscriptionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/SubscriptionSchema"
xmlns:wsol="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/SubscriptionSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
schemaLocation="Schemas/ExpressionSchema.xsd"/>
```

```
  <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
schemaLocation="Schemas/WSOLSchema.xsd"/>
  <xsd:complexType name="subscription">
    <xsd:complexContent>
      <xsd:extension base="wsol:statementType">
        <xsd:sequence>
          <xsd:element ref="wsol:numberWithUnitConstant"/>
          <xsd:element ref="wsol:subscriptionDuration"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```

D.3 Price Default Statement Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/PriceDefaultSchema"
xmlns:priceDefaultSchema="http://www.sce.carleton.ca/~kpatel/WSOL/PriceDefaultSchema"
xmlns:wsol="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/PriceDefaultSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
schemaLocation="Schemas/ExpressionSchema.xsd"/>
  <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
schemaLocation="Schemas/WSOLSchema.xsd"/>
  <xsd:complexType name="priceDefault">
    <xsd:complexContent>
      <xsd:extension base="wsol:statementType">
        <xsd:sequence>
          <xsd:element ref="wsol:numberWithUnitConstant"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```

D.4 Price Statement Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/PriceSchema"
xmlns:priceSchema="http://www.sce.carleton.ca/~kpatel/WSOL/PriceSchema"
xmlns:wsol="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/PriceSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
schemaLocation="Schemas/ExpressionSchema.xsd"/>
  <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
schemaLocation="Schemas/WSOLSchema.xsd"/>
  <xsd:complexType name="price">
    <xsd:complexContent>
      <xsd:extension base="wsol:statementType">
        <xsd:sequence>
```

```
            <xsd:element ref="wsol:numberWithUnitConstant"/>
          </xsd:sequence>
          <xsd:attribute name="service" type="xsd:QName" use="required"/>
          <xsd:attribute name="portOrPortType" type="xsd:QName" use="required"/>
          <xsd:attribute name="operation" type="xsd:QName" use="required"/>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
</xsd:schema>
```

D.5 Penalty Default Statement Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/PenaltyDefaultSchema"
xmlns:penaltyDefaultSchema="http://www.sce.carleton.ca/~kpatel/WSOL/PenaltyDefaultSchema"
xmlns:wsol="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/PenaltyDefaultSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
schemaLocation="Schemas/ExpressionSchema.xsd"/>
   <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
schemaLocation="Schemas/WSOLSchema.xsd"/>
   <xsd:complexType name="penaltyDefault">
      <xsd:complexContent>
         <xsd:extension base="wsol:statementType">
            <xsd:sequence>
               <xsd:element ref="wsol:numberWithUnitConstant"/>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
</xsd:schema>
```

D.6 Penalty Statement Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/PenaltySchema"
xmlns:penaltySchema="http://www.sce.carleton.ca/~kpatel/WSOL/PenaltySchema"
xmlns:wsol="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/PenaltySchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
schemaLocation="Schemas/ExpressionSchema.xsd"/>
   <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
schemaLocation="Schemas/WSOLSchema.xsd"/>
   <xsd:complexType name="penalty">
      <xsd:complexContent>
         <xsd:extension base="wsol:statementType">
            <xsd:sequence>
               <xsd:element ref="wsol:numberWithUnitConstant"/>
            </xsd:sequence>
            <xsd:attribute name="service" type="xsd:QName" use="required"/>
            <xsd:attribute name="portOrPortType" type="xsd:QName" use="required"/>
```

```
            <xsd:attribute name="operation" type="xsd:QName" use="required"/>
          </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:schema>
```

D.7 Additional Penalty Statement Schema

```
<?xml version="1.0"?>
<xsd:schema targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/AdditionalPenaltySchema"
xmlns:additionalPenaltySchema="http://www.sce.carleton.ca/~kpatel/WSOL/AdditionalPenaltySchema"
xmlns:wsol="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/AdditionalPenaltySchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
schemaLocation="Schemas/ExpressionSchema.xsd"/>
    <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
schemaLocation="Schemas/WSOLSchema.xsd"/>
    <xsd:complexType name="additionalPenalty">
      <xsd:complexContent>
        <xsd:extension base="wsol:statementType">
          <xsd:sequence>
            <xsd:element ref="wsol:numberWithUnitConstant"/>
          </xsd:sequence>
          <xsd:attribute name="constraint" type="xsd:QName" use="required"/>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
</xsd:schema>
```

D.8 Management Responsibility Statement Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/ManagementResponsibilitySchema"
xmlns:managementResponsibilitySchema="http://www.sce.carleton.ca/~kpatel/WSOL/ManagementResponsibilityS
chema" xmlns:wsol="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/ManagementResponsibilitySchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
schemaLocation="Schemas/ExpressionSchema.xsd"/>
    <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/WSOLSchema"
schemaLocation="Schemas/WSOLSchema.xsd"/>
    <xsd:complexType name="managementResponsibility">
      <xsd:complexContent>
        <xsd:extension base="wsol:statementType">
          <xsd:sequence>
            <xsd:element ref="supplierResponsibility" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="consumerResponsibility" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="independentResponsibility" minOccurs="0" maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
```

```
    <xsd:element name="supplierResponsibility">
      <xsd:complexType>
        <xsd:attribute name="scope" type="xsd:QName" use="required"/>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="consumerResponsibility">
      <xsd:complexType>
        <xsd:attribute name="scope" type="xsd:QName" use="required"/>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="independentResponsibility">
      <xsd:complexType>
        <xsd:attribute name="scope" type="xsd:QName" use="required"/>
        <xsd:attribute name="entity" type="xsd:anyURI" use="required"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:schema>
```

Appendix E: Service Offerings Dynamic Relationship (SODR) Schema for WSOL 1.1

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/SODRSchema"
xmlns:sodr="http://www.sce.carleton.ca/~kpatel/WSOL/SODRSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:expressionSchema="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/SODRSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by Kruti Patel (Home) -->
   <xsd:import namespace="http://www.sce.carleton.ca/~kpatel/WSOL/Schemas/ExpressionSchema"
schemaLocation="Schemas/ExpressionSchema.xsd"/>
   <xsd:element name="SerOffDynRels">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref="SerOffDynRel" maxOccurs="unbounded"/>
            <!--<xsd:any namespace="##other" processContents="strict" minOccurs="0" maxOccurs="unbounded"/>-
->
         </xsd:sequence>
         <xsd:attribute name="service" type="xsd:QName" use="required"/>
         <!--<xsd:attribute name="targetNamespace" type="xsd:anyURI"/>-->
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="SerOffDynRel">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref="currentSO"/>
            <xsd:element ref="unsatisfiedConstraints"/>
            <xsd:element ref="replacementSO"/>
         </xsd:sequence>
         <xsd:attribute name="sodrName" type="xsd:NCName" use="required"/>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="currentSO">
      <xsd:complexType>
         <xsd:attribute name="name" type="xsd:QName" use="required"/>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="replacementSO">
      <xsd:complexType>
         <xsd:attribute name="name" type="xsd:QName" use="required"/>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="unsatisfiedConstraints">
      <xsd:complexType>
         <xsd:choice>
            <xsd:element ref="unsatisfiedConstraintRef" maxOccurs="unbounded"/>
            <xsd:element ref="expressionSchema:booleanExpression"/>
         </xsd:choice>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="unsatisfiedConstraintRef">
      <xsd:complexType>
         <xsd:attribute name="name" type="xsd:QName" use="required"/>
```

```
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Appendix F: Ontology Schemas Defined for WSOL 1.1

F.1 QoS Metric Ontology Schema

```
<?xml version="1.0"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by Kruti Patel (private) -->
<xsd:schema targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/QoSMetricOntSchema"
xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/QoSMetricOntSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xsd:element name="QoSMetricDefinitions">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="QoSMetricType" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="targetNamespace" type="xsd:anyURI" use="optional"/>
  </xsd:complexType>
  </xsd:element>
  <xsd:element name="QoSMetricType">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsd:NCName" use="required"/>
  </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

F.2 QoS Measure Ontology Schema

```
<?xml version="1.0"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by Kruti Patel (private) -->
<xsd:schema targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/QoSMeasOntSchema"
xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/QoSMeasOntSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xsd:element name="QoSMeasDefinitions">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="QoSMeasUnit" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="targetNamespace" type="xsd:anyURI" use="optional"/>
  </xsd:complexType>
  </xsd:element>
  <xsd:element name="QoSMeasUnit">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsd:NCName" use="required"/>
  </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

F.3 Currency Ontology Schema

```
<?xml version="1.0"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by Kruti Patel (private) -->
<xsd:schema targetNamespace="http://www.sce.carleton.ca/~kpatel/WSOL/CurrencyOntSchema"
xmlns="http://www.sce.carleton.ca/~kpatel/WSOL/CurrencyOntSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xsd:element name="CurrencyDefinitions">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="CurrencyUnit" maxOccurs="unbounded"/>
```

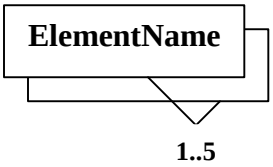
```
      </xsd:sequence>
      <xsd:attribute name="targetNamespace" type="xsd:anyURI" use="optional"/>
   </xsd:complexType>
   </xsd:element>
   <xsd:element name="CurrencyUnit">
   <xsd:complexType>
      <xsd:attribute name="name" type="xsd:NCName" use="required"/>
   </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

Appendix G: Explanation of some symbols used by the XMLSpy Schema Editor to generate schema diagrams

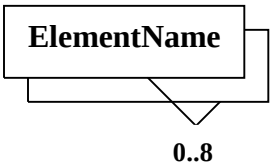
This symbol represents a mandatory single element with MinOcc=1 and MaxOcc=1.

This symbol represents a mandatory single element of type xsd:string, with MinOcc=1 and MaxOcc=1, and that contains parsed character data (#PC-Data).

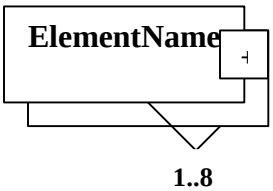
This symbol represents an optional single element with MinOcc=0 and MaxOcc=1.



This symbol represents a mandatory multiple element with MinOcc=1 and MaxOcc=5.

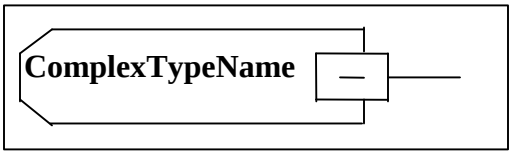


This symbol represents an optional multiple element with MinOcc=0 and MaxOcc= unbounded.

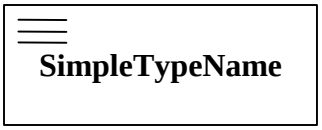


This symbol represents a mandatory multiple element with MinOcc=1 and MaxOcc=unbounded and that contains child elements.

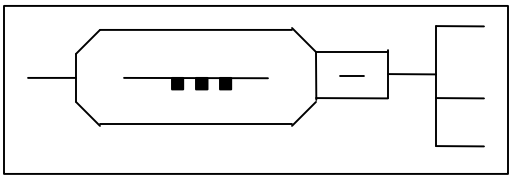
This symbol represents a reference to a global element.



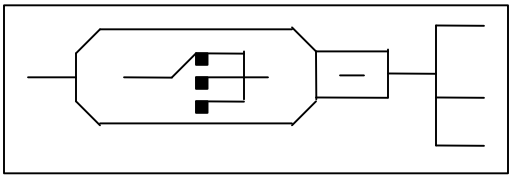
This symbol represents a complex type.



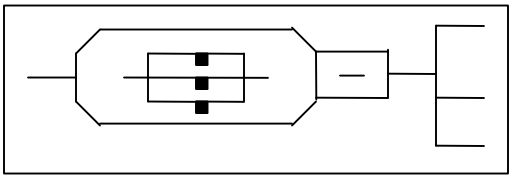
This symbol represents a simple type.



This symbol indicates that the child elements (shown on the right side of this symbol in a schema diagram) have to be specified in a particular sequence.



This symbol indicates that there is a choice between child elements (shown on the right side of this symbol in a schema diagram). Only one child element from the given child elements can be specified at a particular time.



This symbol indicates that all the child elements (shown on the right side of this symbol in a schema diagram) have to be specified, but, they can be specified in any sequence.

This symbol when specified on an element or a complex type indicates that the element or the complex type contains child elements, but, the child elements are not shown in the schema diagram. When this symbol is clicked, the schema diagram expands showing all the child elements and the symbol changes to the one showed below.