Abstract—XML syntax and semantic validations are critical to the correct service transaction specification and service integration based on existing distributed and heterogeneous computing services. Current industry practice of XSLT-based Schematron validation may produce invalid results, and contributes a reusable XML validator component that supports sound integrated syntax/semantic validations and event-driven integration with its environment through public APIs. Ontology of Co-Constraint is about having all the co-constraints expressed within one place instead of being dispersed over multiple Schematron XML Schema documents. This ontology eliminates the semantic heterogeneity and achieves data interoperability by ensuring more flexibility in enterprise data integration.

Keywords—XML; Schematron; co-constraint; syntax validation; semantic validation; integrated validation; co-constraint

I. INTRODUCTION

Valid XML documents are critically important to services computing [7]. The service requests are often in form of XML documents. Web services, the basic communication technology for service access and new service integration based on the existing distributed and heterogeneous services, are based on XML dialects SOAP and WSDL [7].

The service consumer and provider must use the same XML dialect so they could understand each other. An XML dialect specifies the syntax of a class of XML (instance) documents including the supported tag names, element nesting, the supported attributes, and the basic element and attribute data types. DTD and XML Schema (XSD) are the standard schema languages to define XML dialects [2]. XML validating parsers [2], based on either the SAX or DOM framework, can be used to validate whether an XML instance document satisfies the syntax constraints specified in a DTD or XML Schema document.

But in services computing, there are many semantic constraints or co-constraints among the components of an XML instance document that cannot be specified by DTD or XML Schema. For example the value range of an element in an electronic medical record may depend on whether the record is for a male or female patient, and the sales tax rate in an e-commerce transaction depends on the state value for the transaction. Schematron [1] is a popular rule-based XML dialect that allows us to specify such co-constraints for a class of XML documents and then use a standard Schematron validator to validate the co-constraints without coding. Table 1 lists the common co-constraint types supported by Schematron and XSD [4][5].

Over the past decade, the standard implementation of the Schematron validator is to use a standard XSLT style sheet [6][3] to transform a Schematron document into a new validator XSLT style sheet, and then use the latter to validate the XML instance documents, as shown in the following Figure 1.

Table 1 Co-constraints Supported by Schematron and XSD

<table>
<thead>
<tr>
<th>Language Feature</th>
<th>Schematron</th>
<th>XSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sibling content</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Sibling attribute values</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Mutual exclusion</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Element type from attribute presence</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Element type from attribute content</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Attribute type from element content</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Attribute value exclusion</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Abstract Patterns</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
II. SEPARATE SYNTAX AND SEMANTIC VALIDATIONS MAY NOT BE VALID

One important observation of the above XSLT-based implementation of Schematron validation is that it completely separates semantic validation from syntax validation. Now we use a simple counter example to prove that such separate validations may lead to invalid semantic validation results because the information in an XML instance document also includes those defined in the DTD or XML Schema syntax specifications.

Let the following excerpt be part of an XML instance document for e-commerce transactions, and it declares that a transaction has ID value “0120121” and amount “$225.45”:

```xml
<transacs>
  <trans>
    <trans_id>0120121</trans_id>
    <amount>225.45</amount>
  </trans>
  ...
</transacs>
```

Let the following excerpt be part of a DTD document for the XML dialect of the above e-commerce transaction, and it declares that a transacs element includes a sequence of one or more trans elements; each trans element includes elements trans_id, amount, …, in the same order, and has an attribute pay_type that can take on value either “visa” or “master” with the default value being “visa”.

```xml
<!ELEMENT transacs (trans)+>
<!ELEMENT trans (trans_id, amount, …)>  
<!ATTLIST trans pay_type (visa|master) "visa">

Let the following excerpt be part of a Schematron document declaring that the first trans element in a transacs element must have “visa” as the value of its pay_type attribute.

```xml
<rule context="transacs/trans[1]">
  <assert test="@pay_type=’visa’">
    The first trans must have “visa” as its pay type.
  </assert>
</rule>
```

The information in the above XML document actually includes the default value “visa” for attribute pay_type specified in the DTD excerpt above. While this default value is available during syntax validation, it is not available to a Schematron validator if the semantic validation is separate from the syntax validation. Therefore the semantic validation will fail based on the XSLT-based validator. This counter example shows that in general semantic validation separated from syntax validation could be invalid.

III. INTEGRATED SYNTAX AND SEMANTIC VALIDATION THROUGH DOM AND XPATH

In this research we integrate the syntax and semantic validations through a DOM tree [2] which is the output of the DOM-based syntax validation and the input of the XPath-based Schematron validation, as shown in Figure 2.

The DOM validating parser is first used to validate the XML document against its syntax specification in the DTD or XML Schema document, and all information in the XML and DTD/XSD documents is represented in the resulting DOM tree to the left. The same DOM validating parser is also used to validate the Schematron document against the Schematron’s XML Schema specification to ensure that the former is a valid semantic constraint specification and the resulting DOM tree to the right represents the Schematron document. Both of the two DOM trees are fed to our new XPath-based Schematron validator for semantic constraint validation.

IV. NEW FEATURES OF THE INTEGRATED VALIDATOR

In addition to potentially invalid validation results, the XSLT-based Schematron implementation also has several additional drawbacks: (1) the validator result is for people to read thus the validator cannot be easily integrated with other system components; and (2) its functions are limited by the XSLT’s limitations and the latter was not designed for supporting semantic constraint validation. Current integrated validator is designed as a reusable software component based on DOM Level 3 XPath [8]. It supports all key features of Schematron ISO [1] including abstract rules and abstract patterns, network integration through web services, and event-driven loose-coupling. Most importantly, this research provides an open-source framework which serves as a test-bed for new co-constraint types and their efficient validation.

V. CO-CONSTRAINTS AND THEIR PURPOSE

Co-Constraint is a feature of Schematron, which is not possible in XSD and DTD. In general constraints are used in XML to enforce uniqueness of an element or an attribute. A co-constraint is a constraint between two or more values. A co-constraint can exist between data i.e. element-to-element, or element-to-attribute, or attribute-to-attribute. Also, a co-constraint can exist within a single XML document, or across multiple XML documents. The Table 1
is the high level representation of co-constraints, below Figure 3 shows the detailed and specific co-constraint types.

Figure 3 Co-Constraint Types

Following are the common usage of co-constraint in a real time XML document(s) based on Table 1 [9]

### Sibling content
- The content of sibling elements must be the same.
  - **Business Logic:** When both Customer_Id and Cusip are present then they must be equal.
  - **Excerpt XML:**

```xml
<Trade>
  <Customer_Id>CN20395000</Customer_Id>
  <Cusip>CN20395000</Cusip>
  ...
</Trade>
```

### Sibling attribute values
- Sibling elements must have different values for a given attribute.
  - **Business Logic:** Customer_Id and Customer_Nm must have different values.
  - **Excerpt XML:**

```xml
<Trade>
  <Customer_Id>CN20395000</Customer_Id>
  <Customer_Nm>JPMorgan Bank</Customer_Nm>
  ...
</Trade>
```

### Mutual exclusion
- An attribute or a child element must be present, but not both.
  - **Business Logic:** Trade_Id element or Trade_Value element must be present, but not both.
  - **Excerpt XML:**

```xml
<Trade>
  <Trade_Id>0000009239</Trade_Id>
  <Customer_Id>CN20395000</Customer_Id>
  <Trade_Value>RG215</Trade_Value>
</Trade>
```

### Element type from attribute presence
- Element data type dependent on presence or absence of an attribute.
  - **Business Logic:** When Trade_Restricted = "N" then rating must be present.
  - **Excerpt XML:**

```xml
<Trade rating="PAR">
  <Trade_Id>0000009239</Trade_Id>
  <Customer_Id>CN20395000</Customer_Id>
  <Trade_Restricted>N</Trade_Restricted>
  ...
</Trade>
```

### Attribute type from element content
- Attribute values dependent on element content.
  - **Business Logic:** If Cusip is present then rating must not be a numeric value.
  - **Excerpt XML:**

```xml
<Trade type="LMA" rating="PAR">
  <Customer_Id>CN20395000</Customer_Id>
  <Cusip>CN20395000</Cusip>
</Trade>
```

### Attribute value exclusion
- Attributes are required to be different if both are specified.
  - **Business Logic:** Trade type and rating must be different if both are specified.
  - **Excerpt XML:**

```xml
<Trade type="LMA" rating="PAR">
  <Customer_Id>CN20395000</Customer_Id>
  <Cusip>CN20395000</Cusip>
</Trade>
```

Summary of the Co-Constraint Rules are below:

1) When both Customer_Id and Cusip are present then they must be equal.

2) Customer_Id and Customer_Nm must have different values.

3) Trade_Id or Trade Value must be present.

4) When Trade_Restricted = "N" then rating must be present.

5) Presence of rating indicates Trade_Id must be a numeric value.

6) If Cusip is present then rating must not be a numeric value.

7) Trade type and rating must be different if both are specified.
8) When attributes type and rating are not provided in the XML instance document, they should be defaulted to "LMA" for type and "PAR" for rating.

VI. ONTOLOGY USED IN XML SCHEMA

XML documents are syntactic level, and does not support for efficient sharing of conceptualizations shown in Figure 4 [11].

```
| Relational data | serialized to | XML XSD | discover relational structure | OWL |
```

Figure 4 Dataflow diagram

Artificial Intelligence communities are developing Ontologies to promote knowledge sharing and reuse [13]. A common use of ontologies is data standardization and conceptualization via a formal machine-understandable ontology language. For example, the global schema in a data integration system may be an ontology, which then acts as a mediator for reconciliation the heterogeneities between different sources [12].

“A Feasibility Study of Ontology-Based Automatic Document Transformation”, in this research in order to resolve the problem of document transformation, it requires the usage of ontology together with the rules base to perform document transformation. XML schema can specify a structure of any XML document, thus XML document can represent by XML schema. The below Figure 5 shows that the schemas which matches semantically [14].

```
<table>
<thead>
<tr>
<th>Schema Matching Approaches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Individual match approaches</td>
</tr>
<tr>
<td>Combining matches</td>
</tr>
</tbody>
</table>
```

Figure 5 Schema Matching Approaches

Global schema of domain specific or subject developed by document transformation schema matching in the event of schema integration. Ontology represents as the global schema of having the knowledge of source and the target documents [14]. In the above research the ontology was designed to provide a common shared knowledge structure of target and source documents which engaged in the transformation process. Thus, the ontology contains the concepts of target and source documents. The below Figure 6 shown the ontology of frame-based knowledge [14].

```
| Figure 6 The Ontology |
```

“Transforming XML Schema to OWL Using Patterns”, in this research, It was shown the set of patterns which enable the direct automatic transformation from XML schema to OWL. Ivan Bedini et al and Benjamin Nguyen analyzed of XML schema design practices based on B2B standard specifications seen as XML sources and they introduced a detail 40 transformation patterns. Fig 7 shows that this can be applied to a wide set of XML schema sources [15a, b].

```
| Figure 6 XML Schem compnents extraction |
```

D2.4
Hence in this research work, ontology will be implemented by OWL to handling the co-constraints in Schematron XML Schema. Schematron XML schema is nothing but adding co-constraint to XML schema with Schematron standards. There are many ongoing research on XML schema and OWL, thus we are considering several successful methods to transform Schematron XML schema document to OWL. We will write a prototype to select the appropriate method to develop a high level of business (co-constraint) ontology. We will briefly discuss the type of methods next chapter.

VI. METHOD USED IN ONTOLOGY APPROACHES

In this chapter we will discuss the type of approaches to develop the co-constraint ontology. Based on our review, we came up with the following methods that can be appropriate in transforming the Schematron XML Schema to OWL.

1. Map/Lift Schematron XML Schema to OWL
2. Matching Schematron XML Schema and Rule based co-constraint ontologies
3. Using Pattern to develop co-constraint ontology

The result of the above method produces global shared on ontologies.

In order to choose appropriate methodology, it necessary to implement the prototype, i.e. test the methods in appropriate environments.

Map/Lift Schematron XML Schema to OWL

This approach has two folds, which is below

a. Schematron XML document or XML instance document to be translated to RDF
b. Schematron XML schema document to be translated to OWL

Matching Schematron XML Schema and Rule based co-constraint ontologies

In this approach

a. Multiple co-constraint Schematron XML schema documents to be integrated as one co-constraint global schema
b. Global schema which contains the knowledge will be developed as a co-constraint ontology

In order to achieve the above, Protégé has to store the co-constraint ontology. Java Expert System Shell (JESS) is to access and manipulate the co-constraint ontology.

Using Pattern to develop co-constraint ontology

In this method we will be using the existing XML schema patterns developed by Ivan Bedini et al and Benjamin Nguyen. Since the XML schema does not support the co-constraints, thus, the co-constraints pattern to be added.

In this approach

a. Usage of common XML schema patterns to be identified in Schematron XML schema document.
b. Design the Co-Constraints patterns for OWL
c. Merge the identified common XML schema pattern and co-constraint pattern
d. Develop a Global Shared Ontology

All the above methods there may be some limitations based on the requirements, but, that can be resolved by extending the functionalities.

REFERENCES

[10] L Tao and S Golikov, “Integrated Syntactic/Semantic XML Data Validation with a Services Computing”, DPS-Computer Science Department, Pace University, NY, 2013
[12] Isabel F. Cruz Huiyong Xiao, “The Role of Ontologies in Data Integration”, ADVIS Lab, UIC, pages 2-1