Advanced Integrated Syntax and Semantic Validation for Services Computing

Amer Ali and Lixin Tao

Seidenberg School of CSIS, Pace University, White Plains, NY 10606

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. This research proves that the current industry practices of XML validation may produce invalid results or leave costly gaps, and extends a reusable XML validator component that supports sound integrated syntax/semantic validations and event-driven integration with its environment through semantic web technologies and public APIs.

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

I. INTRODUCTION

XML has emerged as by far the best format for data exchange on the internet between heterogeneous platforms and applications [9]. The goal of Services Computing is to make this data exchange more efficient and effective [7]. Services Computing relies heavily on XML to achieve this goal [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].

Valid XML documents are critically important to services computing from business and technology perspective [7]. A National Institute of Standards and Technology commissioned study estimated that members of U.S automotive supply chain alone incur a cost of over $1 billion per year due to interoperability issues [10]. The study states “the greatest component of these costs is the resources devoted to repairing or reentering data files that are not usable for downstream applications.” [10] However, XML on itself is very flexible and provides no inbuilt validation mechanisms [9].

In this paper authors will survey the validation mechanisms based on public APIs available and then extend the integrated XML validation model presented by Tao and Golikov [12].

The first level of validation for an XML instance document is to check whether this document conforms to basic XML specifications such as [2] [11]:

- Start tag must have a matching end tag
- Elements may nest but may not overlap
- There must be exactly one root element
- Attribute values must be quoted
- An element may not have two attributes with the same name

This is basic validation and can be performed with many tools including many web browsers like Internet Explorer and Firefox. The simplest way is to save the file with ‘.xml’ extension and open in the browser window. Most of the XML validation frameworks and tools will generally perform this validation as a first step.

Aforementioned validation is a good start but doesn’t provide any information whether the data is structured in a way that two heterogeneous systems can understand each other. For next level of validation 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 stylesheet [6][3] to transform a Schematron document into a new validator XSLT stylesheet, 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 [12]

<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>

![Diagram of Schematron Validation Using XSLT](image)

II. SEPARATE SYNTAX AND SEMANTIC VALIDATIONS MAY NOT BE VALID

Tao and Golikov [12] made an important observation of the above XSLT-based implementation of Schematron validation that it completely separated semantic validation from syntax validation. They used 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 specifications. Below are the details of their observation and example.

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
<html>
  <transacs>
    <trans>
      <trans_Id>0120121</trans_Id>
      <amount>225.45</amount>
    </trans>
  ....
</transacs>
</html>
```

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

To fix this validation issue they proposed below model. In their research they integrated 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 XML DTD/XSD. The Schematron validator then uses the DOM parser's result of the XML document to validate the constraints specified in the Schematron documents.

![Diagram of Integrated Syntax/Semantic Validation](image)
resulting DOM tree to the right represents the Schematron document. Both of the two DOM trees are fed to new XPath-based Schematron validator for semantic constraint validation [12].

IV. EXTENDING SEMANTIC VALIDATION CAPABILITIES
Tao and Galikov model bridges an important gap in XML validation. But imagine a scenario where an upstream system/application enters a different pay_type other than ‘visa’ or ‘master’. The validator developed so far will plainly reject the transaction. But is this rejection a valid business rejection? It may not be so! Let’s go back to Services Computing. Visa Inc. is one of the beneficiaries of this computing paradigm. They offer service in over 150 countries and their computing services integrate with millions of direct and indirect service providers [13]. They constantly integrate with new systems and do their business with different brand names or alliance names in different regions. See table 2 for some examples of the brand names VISA cards are also known with. Say for instance, an online store in UK still uses old/alternative name “delta” instead of “visa”. They send their transaction to our validator with pay_type as “delta”. Our validator will plainly reject the transaction as it is not intelligent enough to know that “delta” is an alternative name for visa. This is a perfectly valid business transaction but the validator has rejected it. As we saw earlier [10] that reentering and repairing data documents incur significant costs. Is there a way we can recover from this situation without rejecting the transaction and going back to source system? One way is to update our DTD document to include “delta” as an option like below:

![DTD snippet](image)

But this approach will require us to update all the producing and consuming systems every time we have to add a similar option. This may not be an issue if we have only few systems but global systems like Visa with millions of upstream/downstream producers and consumers will require immense resources.

Table 2 Possible brand names that may point to concept “Visa” in financial services domain

<table>
<thead>
<tr>
<th>Region</th>
<th>Possible keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>US, Worldwide</td>
<td>Visa</td>
</tr>
<tr>
<td>UK</td>
<td>Delta</td>
</tr>
<tr>
<td>US, Worldwide</td>
<td>Plus</td>
</tr>
<tr>
<td>Worldwide except US</td>
<td>Electron</td>
</tr>
<tr>
<td>Australia</td>
<td>EFTPOS</td>
</tr>
</tbody>
</table>

Table 3 Semantics of “Visa” in different domains

<table>
<thead>
<tr>
<th>Domain</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Government</td>
<td>Visa - A document to enter or leave a country</td>
</tr>
<tr>
<td>Commerce</td>
<td>Visa Inc - a financial services company</td>
</tr>
<tr>
<td>Commerce</td>
<td>VISA - A Mexican beverage company</td>
</tr>
<tr>
<td>Geography</td>
<td>Visa - A river in Romania</td>
</tr>
<tr>
<td>Transportation</td>
<td>Citroen Visa - an auto mobile</td>
</tr>
<tr>
<td>Entertainment</td>
<td>V.I.S.A - a French record label</td>
</tr>
<tr>
<td>Computer Science</td>
<td>VISA - Virtual Instrument Software</td>
</tr>
</tbody>
</table>

V. ADVANCED VALIDATOR TO USE SEMANTIC WEB TECHNOLOGIES
This paper proposes one way of salvaging such validation issues without rejecting valid business transactions. We propose to extend the validator model to include an ontology based validator leveraging Semantic Web technologies like RDF, RDFS and OWL. This semantic validator will be called as a last resort if either schema based validation (XSD/DTD) or rule based validation (Schematron) fails. This will add immense power of semantic web technologies to the existing model. Let’s visit our example of visa/delta again. Assuming our domain is already semantic enabled and has well defined ontology, all we would need is to declare that “visa” and “delta” are equivalent classes. Let below excerpt be part of an ontology document defining the concept “visa”:

![Ontology snippet](image)

Now assuming www.deltaCard.com has already defined “delta” concept that has meanings similar to “visa” in this particular domain. All you have to do is use “owl:equivalentClass” property in your earlier definition as below excerpt:

![Ontology snippet](image)
This new model will have several advantages over the existing model (1) New model will try to understand the meanings of the data and not just its structure thus reducing validation failures (2) it will provide scalability to Services Computing consumers and producers by making the integration relatively simpler (3) the end users and intermediaries can also share the burden of integration with service providers by adding to semantic definitions of key concepts.

VI. NEW FEATURES OF THE ADVANCED INTEGRATED VALIDATOR

The new validator model retains all the functionality of the Tao and Golikov model such as reusable components based on DOM Level 3 and XPath. It also retains all the key features of Schematron ISO including but not limited to abstract rules and abstract patterns, cross platform integration through web services and flexible even-driven loose-coupling [12]. In addition to the retained features the new advanced model also adds the power of semantic web technologies like OWL. It provides the ability to look beyond just the keyword and look at the meanings of the data and try to recover from validation failures for critical business transactions thus potentially saving important resources and help achieve the Services Computing its goal of efficient and effective services. It also provides new features such as class inheritance, cardinality, intersection, union, complement and equivalence. It also provides symmetric, transitive and inverse properties [9]. All these features combined with earlier model enable this research to provide an open-source framework which serves as a test-bed for new co-constraint types, their efficient validation and leverages the power of semantic web technologies.

REFERENCES