What is the Overlap between XML Schema 1.1 vs. XML Schema 1.0 + Schematron?

Roger L. Costello
August 2012

XML Schema 1.1 has twenty one new capabilities.

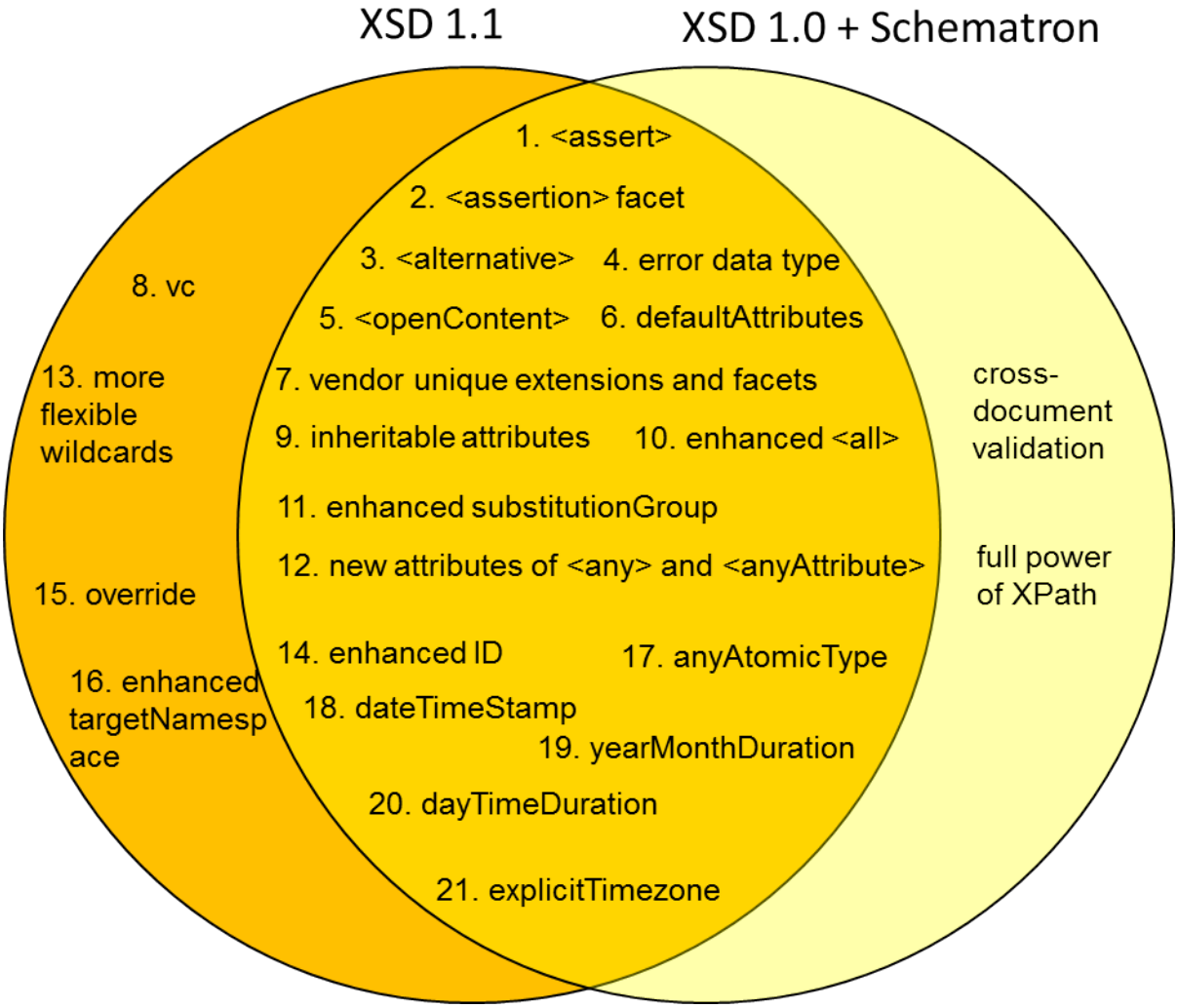
How do those new capabilities stack up against a tandem of 1.0 and Schematron? Can the new capabilities be implemented using the existing 1.0 schema technology plus Schematron?

It is an important question. Organizations are well entrenched in their 1.0 XSDs plus their Schematron schemas. They have invested heavily in these technologies. And they have expertise in these technologies. And there is good support for these technologies.

So what is the benefit of switching to XSD 1.1? Will it provide more capabilities than the existing tandem of XSD 1.0 and Schematron?

This paper attempts to shed some light on these questions.

Here is a Venn Diagram. The left circle is XSD 1.1. The right circle is XSD 1.0 plus Schematron. The new capabilities in XML Schema 1.1 are shown. If the new capability can be equivalently expressed using a tandem of XSD 1.0 plus Schematron then it is in the intersection. If the new capability cannot be equivalently expressed then it is to the left, exclusively in the XSD 1.1 circle. Capabilities that are unique to XSD 1.0 plus Schematron are to the right, exclusively in the XSD 1.0 plus Schematron circle.



The following table shows in the left column the new capabilities in XML Schema 1.1 and in the right column it describes how to achieve the equivalent capability using a tandem of 1.0 plus Schematron. If the 1.1 capability cannot be achieved using the tandem then it says so.

	New capability provided by XML Schema 1.1	**How to achieve the equivalent capability using a tandem of XML Schema 1.0 plus Schematron**
1.	<assert>	This capability is readily achieved using Schematron. In fact, Schematron provides more capability than does the <assert> element because <assert> is limited just to one XML document, whereas Schematron can make assertions across multiple XML documents.
2.	<assertion> facet	Again, Schematron provides this capability, plus more.
3.	<alternative>	The equivalent capability can be achieved by the 1.0 schema declaring an element of type, xs:anyType, and then

		the Schematron schema contains the rules, “If the value of the attribute is A then the content of the element must be xyz; if the value of the attribute is B then the content of the element must be wxy; and so forth”
4.	error data type	Readily achieved using a Schematron rule that checks for erroneous data.
5.	<openContent>	This can be achieved by adding <any> elements everywhere open content is desired. This is not as clean as the <openContent> element provided in XSD 1.1.
6.	defaultAttributes	The equivalent capability can be achieved by the 1.0 schema declaring anyAttribute in each complexType and then in Schematron have rules that specify the allowable default attributes.
7.	vendor unique extensions and facets	The equivalent capability can be achieved by the 1.0 schema declaring an element of type string and then in Schematron specify the rules for the extensions and facets.
8.	vc:minVersion, vc:maxVersion, vc:typeAvailable, vc:typeUnavailable, vc:facetAvailable, vc:facetUnavailable	These cannot be achieved using a tandem of XSD 1.0 and Schematron. However, vc is probably not useful until there are more versions of XML Schema.
9.	inheritable attributes	Readily achieved using a Schematron rule that has assertions which take into account the value of attributes anywhere in the XML document, even across XML documents.
10.	enhanced <all> element	Readily achieved by using a repeatable <choice> and then using Schematron rules to enforce cardinality.
11.	enhanced substitutionGroup	Readily achieved by replacing the substitutionGroup with an XSD 1.0 <choice> element.
12.	new attributes of <any> and <anyAttribute>	Readily achieved through the use of Schematron rules.
13.	more flexible rules for wildcards	This cannot be achieved using a tandem of XSD 1.0 and Schematron.
14.	element can have multiple attributes of type ID and the ID type can have a fixed or default value	Declare multiple attributes to be of type string and then use Schematron to enforce uniqueness.
15.	<override>	This cannot be achieved using a tandem of XSD 1.0 and Schematron.
16.	enhanced targetNamespace enables a complexType to restrict a complexType in another namespace	This cannot be achieved using a tandem of XSD 1.0 and Schematron.
17.	anyAtomicType	Declare an element to be of type anyType, and then apply Schematron rules to ensure the element’s value is

		anyAtomicType.
18.	dateTimeStamp	Declare the item to be of type dateTime and then use Schematron to verify the dateTime value has a time zone value.
19.	yearMonthDuration	Declare the item to be of type duration and then use Schematron to verify the duration value contains just years and months.
20.	dayTimeDuration	Declare the item to be of type duration and then use Schematron to verify the duration value contains just day and time.
21.	explicitTimezone facet	Use Schematron to check that the value specifies a timezone.