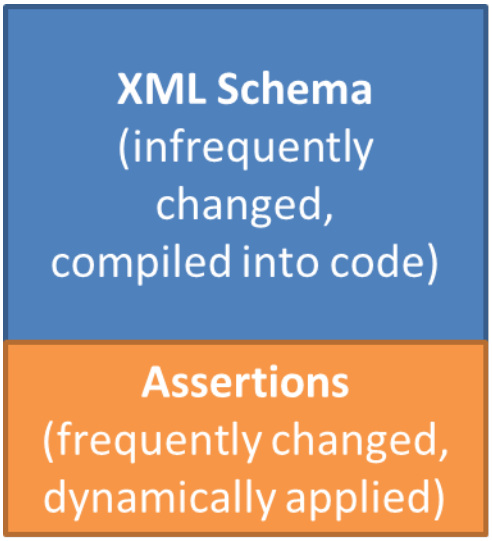
Rick Jelliffe on Designing XML Schemas in a way that Support Parts which Frequently Change

Rick Jelliffe recently made a keen observation on how to design schemas in a way that they support parts which frequently change [1]:

> XSD schemas tend to be compiled into code, rather than dynamically used. So a change to the schema can be a big deal, especially after deployment. A schema change often means a new release of the application or database.
>
> So it isn't so much whether there are two files, but whether there are two levels to allow agility. In XSD, you could get the effect by, say, having a base schema that changes rarely and is used for compilation, and a derived schema that has the asserts in it, and is loaded dynamically.

Here is a graphic that illustrates the two levels that Rick mentions:



Let's take an example to get a firm grasp on Rick's ideas.

Example: An XML Schema is created for a BookStore. Here is a sample instance document:

```
<?xml version="1.0"?>
<BookStore>
    <Book>
        <Title>Conceptual Mathematics</Title>
        <Author>F. William Lawvere</Author>
        <Cost currency="USD">49.73</Cost>
    </Book>
    ...
</BookStore>
```

Let's assume that the requirement for providing the title of the book, the author, and a cost is long-lived and unchanging. Thus it is appropriate to create an XML Schema for implementing that requirement:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema">

  <element name="BookStore">
    <complexType>
      <sequence>
        <element name="Book" maxOccurs="unbounded">
          <complexType>
            <sequence>
              <element name="Title" type="string" />
              <element name="Author" type="string" />
              <element name="Cost">
                <complexType>
                  <simpleContent>
                    <extension base="decimal">
                      <attribute name="currency" type="string" />
                    </extension>
                  </simpleContent>
                </complexType>
              </element>
            </sequence>
          </complexType>
        </element>
      </sequence>
```

```
    </complexType>
  </element>

</schema>
```

That XML Schema is compiled into code and an application is built to process XML instances conforming to the XML Schema.

Next, the book store regularly has sales. For example, today there is a 10% discount on all books, last week there was a 5% discount. Thus sales information is dynamic and should be in a separate layer from the static XML Schema.

I'm Confused

Okay, the sales information is dynamic and should be expressed as assertions in a separate layer and the assertions should be dynamically loaded.

What does that mean?

Does it mean that assertions should be used to modify the values of the <Cost> elements in XML instances? How can that be? Assertions don't modify anything. They just check for validity. So it can't mean this.

Perhaps it is a bad example and does not illustrate the kind of thing that should be dynamic? If so, what is a good example?

What does it mean to dynamically load assertions?

Reference

[1] See Rick Jelliffe's post to the xml-dev list:
http://lists.xml.org/archives/xml-dev/201208/msg00047.html