Service Modeling Language (SML)

Pratul Dublish
Principal Program Manager
Microsoft Corporation

Outline

- Introduction to SML
  - SML Model
- Inter-Document References
  - URI scheme
  - deref() extension function
  - Schema-based constraints
  - Identity constraints
- Rules (Schematron constraints)
- SML Model Validation

What is SML?

- SML is an extension of XML Schema 1.0 with
  - Inter-document references
  - Schema-based constraints on inter-document references
    - `sml:acyclic`
    - `sml:targetElement`
    - `sml:targetType`
    - `sml:targetRequired`
  - Identity constraints across references
    - `sml:key`
    - `sml:unique`
    - `sml:keyref`
  - `deref()` XPath extension function for obtaining the target element of a reference
  - Rules for expressing rich constraints (Schematron)
  - Binding of rules to schema and/or instance documents

Profiles Defined by SML

- XML Schema 1.0
  - Very broad profile - almost all valid XML schemas are valid SML schemas
  - Not supported
    - `xs:redefine`
    - Unqualified local elements
  - Limited Support
    - Target namespace must be specified
- Schematron
  - All elements and attributes are supported
  - Limited Support
    - If the `queryBinding` attribute is specified, it must be set to “xpath1.0”

SML Model

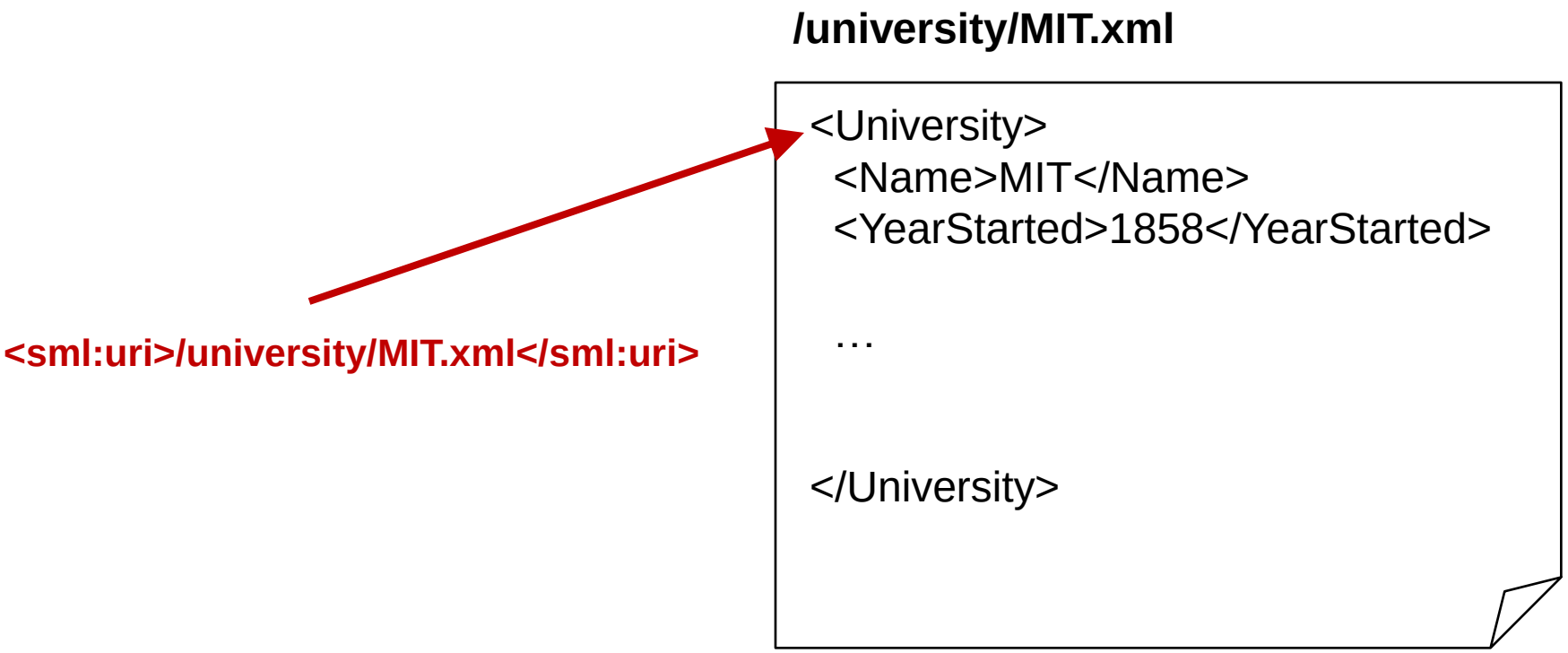
- An **SML model** is a *set* of inter-related XML documents. It consists of two disjoint subsets
  - **Definition** Documents: The subset that describes the schemas and rules that govern the structure and content of the model's documents
    - Schema Documents: Conform to SML's profile of XML Schema 1.0
    - Rule Documents: Conform to SML's profile of Schematron
  - **Instance** Documents: The subset that describes the structure and content of the modeled entities.
- Model Validation: The process of verifying that *all* documents in a model are valid with respect to the model's definition documents.

Inter-Document Reference

- Allows one XML document to reference *an element* in another document
  - Can also be used to reference an element in the same document
- Used to express relationship/dependency between two XML documents. E.g.,
  - The document for a university may reference the documents for its enrolled students
- SML *does not* mandate the use of any specific scheme for references and allows implementations to choose suitable schemes for references
  - But it does defines how references must be represented using URI and EPR schemes
    - `sml:uri` element for URI scheme
    - `wsa:EndpointReference` element for EPR scheme

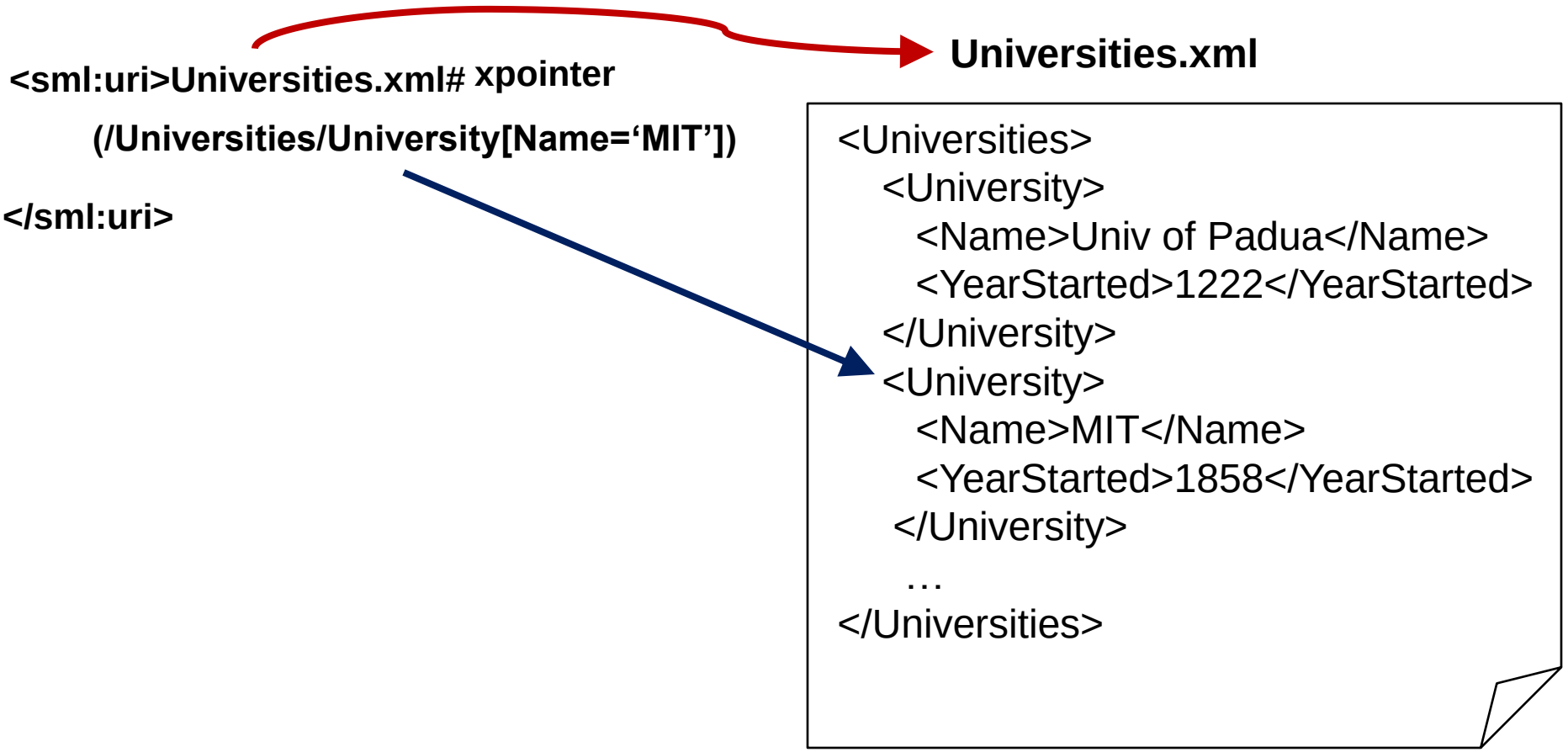
URI Scheme for References

An inter-document reference whose value is a document URI references the **root element** of the document



URI Scheme for References

An inter-document reference whose value is a document URI suffixed with an XPointer expression references the element that results from evaluating the XPointer expression in the context of the referenced document



Identifying References

- The global attribute – `sml:ref` – is used to identify references
  - References can be identified without PSVI
- A reference element must have `sml:ref="true"`
  - Its child *elements* may contain a reference represented in *one or more* schemes

```
<EnrolledCourse  sml:ref="true" >
    <sml:uri>
      /Universities/MIT/Courses.xml#xmlns(u=urn:university)
      xpointer(/u:Courses/u:Course[u:Name='PHY101'])
    </sml:uri>
    <wsa:EndpointReference>
      <wsa:Address>http://www.university.example</wsa:Address>
      <wsa:ReferenceParameters>
        <University>
          <Name>MIT</Name>
        </University>
        <Course>
          <Name>PHY101</Name>
        </Course>
        </wsa:ReferenceParameters>
      </wsa:EndpointReference>
  </EnrolledCourse>
```

`deref()` Extension Function

- XPath 1.0 extension function for de-referencing reference elements, i.e., obtaining the target elements for a set of reference elements
  - Input: Node set of elements
  - Output: Node set of elements that are the target of *reference elements* in the input node set
- Enables the definition of constraints/rules that span multiple documents since XPath expressions can use `deref()` to traverse a reference from its source to its target

sml:refType

- Base type for references
- Must be used for specifying schema-based constraints on references

```
<xs:complexType name="refType" sml:acyclic="false">
   <xs:sequence>
      <xs:any namespace="##any" minOccurs="0"
              maxOccurs="unbounded"
              processContents="lax"/>
   </xs:sequence>
   <xs:attribute ref="sml:ref" use="required"  fixed="true" />
   <xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>
```

Schema-based Constraints on References

- All schema-based constraints on references (except `sml:acyclic`) can only be specified on element declarations whose type is `sml:refType` or a derived type of `sml:refType`

```
<xs:element name="EnrolledCourse"
            sml:targetType="tns:CourseType"
            type="sml:refType">
```

- `sml:acyclic` can only be specified on derived types of `sml:refType`

```
<xs:complexType name =“PreRequisiteType" sml:acyclic=“true">
  <xs:complexContent>
    <xs:extension base="sml:refType" />
  </xs:complexContent>
</xs:complexType>
```

Schema-based Constraints on References

- `acyclic`: instances of a reference type can not result in cycles in valid SML models
  - **Scenario**: A reference type that represents pre-requisite relationship between courses in a university
- `targetType`: specifies the type of a reference's target element
  - **Scenario**: The type of the element referenced by a student reference in a university must be `StudentType`
- `targetElement`: specifies the global element declaration (GED) of a reference's target element
  - **Scenario**: The element referenced by an enrolled course reference must be an instance of the GED `Course`
- `targetRequired`: specifies that a reference's target element must be present in the model
  - **Scenario**: The element referenced by an enrolled course reference must be must be present in the model

Schema-based Constraints on References

```
<xs:complexType name="StudentType">
      <xs:sequence>
         <xs:element name="ID" type="xs:string"/>
         <xs:element name="Name" type="xs:string"/>
         <xs:element name="EnrolledCourses"
                        minOccurs="0“
                        maxOccurs=“unbounded”
                        sml:targetElement="tns:Course"
                        sml:targetRequired="true"
                        type="sml:refType">
    </xs:sequence>
</xs:complexType
```

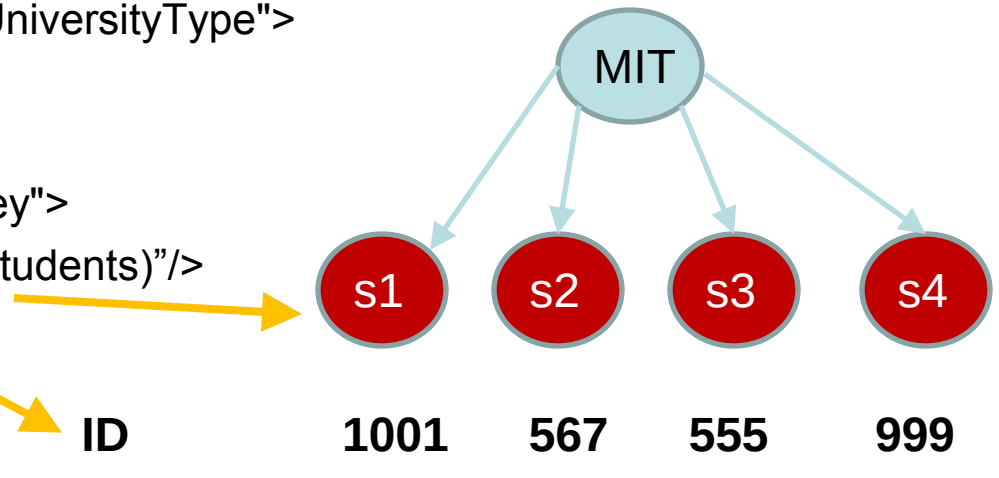
Target element must be an instance of the global element declaration tns:Course

Target element must be present in the model

Key/Unique Constraints across References

- `sml:key`, `sml:unique`, and `sml:keyref`
  - Extend `xs:key`, `xs:unique`, and `xs:keyref` to inter-document references
  - **Scenario**: A university document has references to documents corresponding to students enrolled in the university. All students in the university must have an ID and the ID must be unique

```
<xs:element name="University" type="UniversityType">
  <xs:annotation>
    <xs:appinfo>
      <sml:key name="StudentIDisKey">
        <sml:selector xpath="deref(Students)"/>
        <sml:field xpath="ID"/>
      </sml:key>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
```



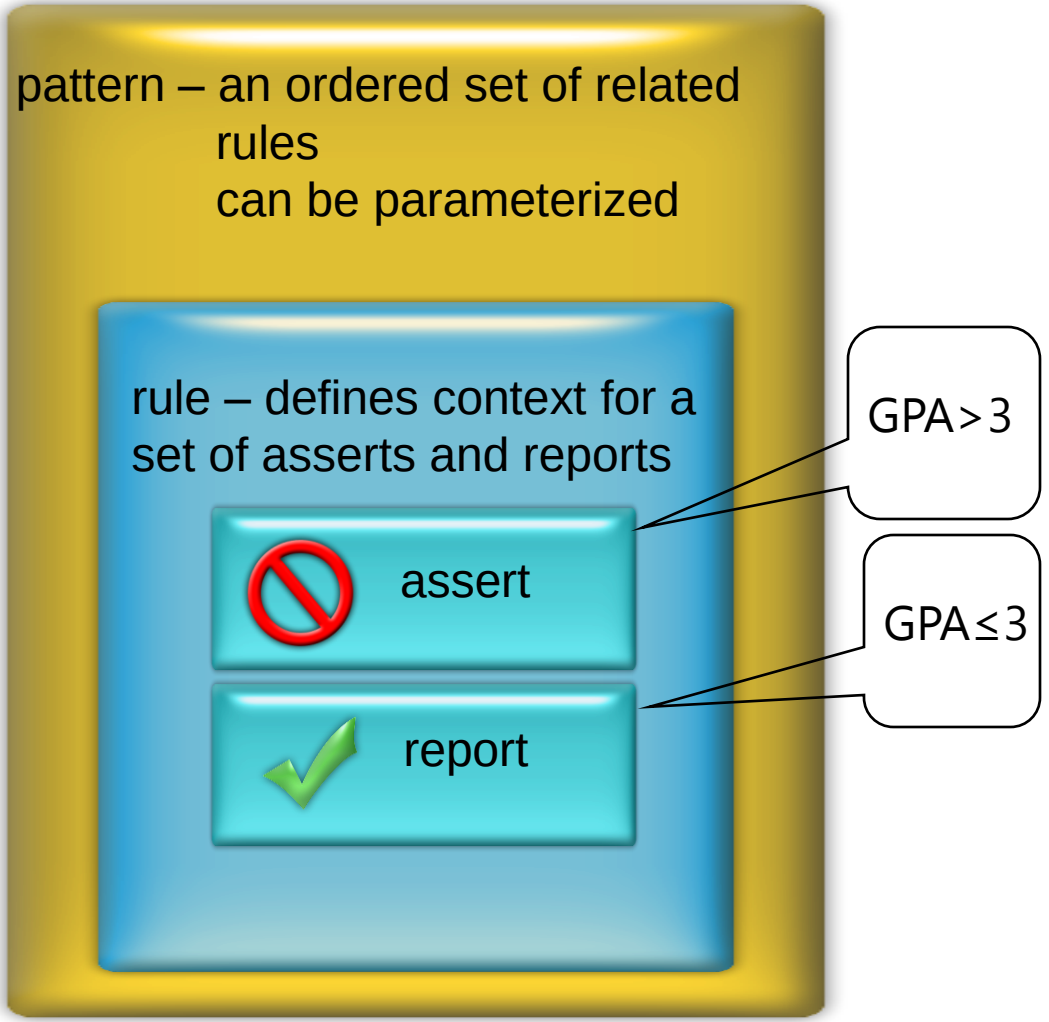
sml:keyref

```
<xs:element name="University" type="UniversityType">
    <xs:annotation>
      <xs:appinfo>
        <sml:key name="StudentIDisKey">
          <sml:selector xpath="deref(Students)"/>
          <sml:field xpath="ID"/>
        </sml:key>
        <sml:keyref name="CourseStudent"
                    refer="StudentIDisKey">
            <sml:selector xpath="deref(deref(Courses)/EnrolledStudents)"/>
            <sml:field xpath="ID"/>
        </sml:keyref>
      </xs:appinfo>
    </xs:annotation>
</xs:element>
```

Schematron

- An ISO standard (ISO/IEC19757-3) for defining assertions on XML documents
- Uses XPath 1.0 as the default query binding
  - A conforming implementation can support other query languages such as XPath2.0, XQuery, XSLT, etc.
- SML uses XPath 1.0 as the query binding
  - Augmented with `deref()` extension function

Schematron


pattern – an ordered set of related rules
can be parameterized
rule – defines context for a set of asserts and reports
assert
report
GPA>3
GPA≤3

Associating Patterns with Documents

- Schematron does not define any mechanism for specifying the pattern(s) that need to be evaluated for a set of XML documents or elements
- SML associates patterns embedded in the `xs:appinfo` child element of a *type definition* or *global-element definition* with **all** elements that are instances of the type or global-element definition
- Implementation are required to provide a mechanism to bind Schematron patterns that are authored in separate documents, i.e., not embedded in schema, to documents in a model
  - Implementations are free to choose a suitable mechanism
  - SML IF defines the following mechanism

```
<ruleBinding>
    <DocumentAlias>/university</DocumentAlias>
    <RuleAlias>/rules/academic/univ.xml</RuleAlias>
</ruleBinding>
```

Schematron Example

An IPv4 address must have 4 bytes
An IPv6 address must have 16 bytes

```
<xs:complexType name="IPAddress">
    <xs:annotation>
        <xs:appinfo>
            <sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron">
              <sch:pattern id="IPAddressPolicy">
                <sch:rule context=".">
                        <sch:assert test="version != 'V4' or count(address) = 4">
                            A v4 IP address must have 4 bytes.
                        </sch:assert>
                        <sch:assert test="version != 'V6' or count(address) = 16">
                            A v6 IP address must have 16 bytes.
                        </sch:assert>
                </sch:rule>
              </sch:pattern>
            </sch:schema>
        </xs:appinfo>
    </xs:annotation>
    <xs:sequence>
        <xs:element name="version" type="xs:string" />
        <xs:element name="address" type="xs:byte" minOccurs="4" maxOccurs="16" />
    </xs:sequence>
</xs:complexType>
```

Schematron Constraints Across References

```
<xs:element name="StrictUniversity" type="tns:UniversityType">
  <xs:annotation>
    <xs:appinfo>
        <sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron">
          <sch:ns prefix="u" uri="urn:university" />
          <sch:ns prefix="smlfn"
                  uri=" http://schemas.serviceml.org/smlfn/query/2006/07"/>
          <sch:pattern id="StudentPattern">
            <sch:rule context=".">
              <sch:assert test="count(smlfn:deref(u:Students)) =
                                count(smlfn:deref(u:Students)[starts-with(u:ID,'99')])">
                One/more students have an ID that does not begin with 99
              </sch:assert>
            </sch:rule>
          </sch:pattern>
        </sch:schema>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
```

Model Validation

- An **SML model** is a *set* of inter-related XML documents. It consists of two disjoint subsets
  - **Definition** Documents: XML Schema 1.0 documents and Schematron documents used for model validation
  - **Instance** Documents
- Model validation verifies that *all* documents in a model are valid with respect to the model's definition documents
- Each document in the model must be XML Schema valid with respect to the XML Schema documents in the model's definition
- Each document in the model must satisfy all applicable Schematron constraints specified in the model's definition
  - Embedded in some schema document in the model's definition
  - Defined in separate Schematron documents
- Each document in the model must satisfy all applicable `sml:target*` constraints
- The model must not contain a cycle whose edges are references of type **R** if **R** is an acyclic reference type

Use of SML

- SML is domain-neutral and can be used in any domain/scenario where XML Schema can be used
- SML can be used to build rich XML Schema based models of complex services and systems where one/more of the following is true
  - A single XML document representation of the modeled entity is not feasible or practical
  - Arbitrary XPath expressions are required to specify constraints on some aspects of the modeled entity's structure/content
    - Can't be specified using the schema-based constraints in XML Schema 1.0
  - There is a need to specify some constraints outside of the modeled entity's schema definition