Schema Binding Proposal

Sandy Gao
Valentina Popescu

1 Terminology

Schema document: an <xs:schema> element; can be an XML fragment

Schema: a set of schema components; a schema is normally (but not required to be) constructed from one or more schema documents

Schema component: an element declaration or a type definition or a particle or …

Include: A schema document can include another schema document using <xs:include>. Both schema documents contribute to the same schema; and both correspond to schema components from the same target namespace (or no namespace). If the included schema document does not have a target namespace, namespace of the including schema document is used.

Redefine: Similar to include, but use <xs:redefine>, and the redefining schema document can replace certain included components with new components.

39	**_Import_**:	Allows the importing schema document to refer to components from the imported
40		namespace (or no namespace), which must be different from the importing
41		schema document's target namespace. If the combination of the "namespace"
42		attribute and the "schemaLocation" attribute on <xs:import> resolves to a schema
43		document, then the resulting schema also includes components from the
44		imported schema document.

45 **_Schema composition_**: (In this document) construct a single schema from multiple schema documents,
46 using the above include, redefine and/or import mechanisms.

47 **Note**: "a schema" is **_not_** equal to "a schema document"!

2 Problem definition

49 In performing SML model validation over the SML model packaged in an SML-IF instance, associations
50 between XML Schema definition documents and instance documents need to be drawn, both to
51 completely validate XML Schema documents themselves (to make sure they produce valid schemas) and
52 to establish schema-validity of the instance documents.

53 Schema documents can be connected with other schema documents using composition features
54 provided by XML Schema. This includes <xs:include>, <xs:redefine>, and <xs:import>. A schema
55 document's validity may depend on other schema documents it includes/redefines/imports, or even other
56 schema documents that include/redefine/import it.

57 When validating a model instance document, a precise list of schema documents need to be associated
58 with it for a "schema" and the instance document is schema-assessed using this schema.

59 The XML Schema 1.0 specification provides more flexibility in constructing the schema used for
60 assessment than is appropriate for the semantics defined by SML and SML-IF validation:

61 • It allows processor latitude in terms of locating schema documents (resolving namespace and
62 schema location attributes) and composing schema documents together to form a single schema.

63 • Schema location attributes can be ignored in some cases ("xsi:schemaLocation" in instance
64 documents and "schemaLocation" on <xs:import>); and allowed to "fail to resolve" in others
65 ("schemaLocation" attribute on <xs:include> and <import>). Known schema and SML
66 implementations behave differently with respect to how/whether they process schema location
67 attributes.

68 • Multiple imports of the same namespace allow all but the first one to be ignored.

69 So it is clear that we have no hope of guaranteeing general case interoperability using anything based
70 only on XML Schema given the constraints above, and SML-IF needs to specify how to determine such
71 associations.

72 NOTE: this proposal is only about SML model validation, and not SML-IF validation (against the IF
73 schema). Unless otherwise indicated, "validation/validity" in the following sections is always about SML
74 model validation.

3 Requirements

3.1 Support schema composition

77 There are many real-life schemas that are constructed from multiple schema documents. Such schemas
78 may span multiple namespaces (hence the need for import); components from each namespace may be
79 further divided into multiple schema documents (hence the need for include).

80 Schema has a feature often referred to as "chameleon include". This means that a schema document
81 with a target namespace includes or redefines another schema document without a target namespace,
82 and the result is as if the included/redefined document had a target namespace that's the same as the
83 including/redefining document. SML-IF needs to support this usage scenario.

3.2 Support schema versioning

85 Schema authors can't anticipate how their schemas will be used, hence the need to evolve schemas.
86 There are different versioning scenarios. There are cases where minor modifications of older versions
87 suffice, and redefine can be used. Some schemas need to be rewritten to accommodate new
88 requirements, and new namespace may or may not be introduced (compatibility is often a good reason
89 for not changing namespaces). There are also cases where there are generic and specific versions (as
90 opposed to previous and next versions), which often co-exist and share the same namespace.

91 To support this, SML-IF needs to be able to package in the same SML-IF instance different versions of
92 the same schema in the same namespace.

3.3 Deterministic

94 For a given SML-IF instance, there MUST be no ambiguity in determining how schema documents (that
95 are included in this instance) are connected using <xs:include>, <xs:redefine>, and <xs:import>, and
96 therefore MUST be no ambiguity in determining which schema documents are used to form a schema
97 against which a given instance document is validated.

3.4 Full schema support

99   Being a generic validation language, SML supports all schema features. Being a mechanism to transmit
100 SML models, SML-IF also needs to support full schema features, especially <xs:include>, <xs:redefine>,
101 and <xs:import>. For example, in an SML model, if an instance document I is validated against a schema
102 formed from a schema document A, which redefines schema document B, then it MUST be possible to
103 transmit I, A, and B in an SML-IF instance and maintain their relationship.

3.5 Schema document exchange

105 An SML-IF document can contain XML Schema documents within its definition documents that are
106 attached for exchange purposes only. These documents are not intended to be used for XML Schema
107 validity assessment of the model instance documents. SML-IF needs to support this use case and
108 ensure that documents of this purpose do **not** participate in model instance document validation.

109 This is analogous to the case we already have for rule documents, except rule documents do not have a
110 "bind to all" default as we are contemplating for XML Schema documents. Any new types of definition
111 documents added in the future will have to address similar concerns, whose syntax will be influenced by
112 the default binding (all or none).

113 Note that for both schema documents and Schematron rule documents that are not bound to any
114 instances, their validity should still be checked when assessing SML model validity, as required by SML,
115 which has:

116 • Each XML Schema document in the model's definition documents MUST satisfy the conditions
117   expressed in Errors in Schema Construction and Structure (§5.1). [XML Schema Structures]

118 • Each Schematron document in the model's definition documents MUST be a valid Schematron
119   document [ISO/IEC 19757-3]

120 4 Constraints

121 4.1 Support access to schema documents outside of SML-IF

122 We do not want to force all schemas necessary to validate the model instance documents packaged by a
123 single SML-IF instance to be included by value in every SML-IF instance. It is not clear this would even
124 be sensible in a repository interchange scenario, let alone the more general case of usage scenarios
125 some have mentioned for SML-IF like web services message exchanges.

126 4.2 Ignorable schema locations

127 We cannot require honoring of xsi:schemaLocation and xsi:noNamespaceSchemaLocation in instance
128 documents or schemaLocation on <xs:import>, because

129 • Some existing implementations ignore them

130 • Honoring schema location in instance documents may have security consequences

131 Schema specification does require that processors attempt to resolve schema locations specified on
132 <xs:include> and <xs:redefine>. It is not an error for such attempt to fail for <xs:include>. It is an error
133 when <xs:redefine> contains non-annotation content.

134 It's more flexible for <xs:import>. Schema allows any strategy for processors to locate components to
135 import, based on either or both of the namespace and the schema location.

136 4.3 Include definition and instance documents as-is

137 SML-IF instance producers may not have control over the content of the schemas necessary for
138 validation of model instance documents, where "control" means what is coded in the files. I.e. there will
139 be cases where xs:import and xs:include are coded, with and without schemaLocation, and multiple files
140 containing schema components for the same namespace will be observed.

141 4.4 Lazy schema assembly

142 Schema specification allows schemas to be assembled lazily. A partial schema can be used to validate
143 an instance document, and more components can be added to the schema during the validation, as long
144 as the new components don't change the validation result of information items that are already validated.

145 This is sometimes not easy to enforce, but a consequence of "supporting full schema" implies that SML-IF
146 validation cannot violate this constraint.

147 4.5 Support reference constraints

148 Reference-related constraints (targetElement, targetType, acyclic, SML identity constraints) need to be
149 properly supported. When 2 documents **A** and **B** are connected by an SML reference, these constraints
150 require the ability to determine whether a component from the schema used to assess **A** is identical to a
151 component from the schema used to assess **B**. The schema spec doesn't define identity of components
152 across multiple schemas. The same source declaration may produce totally different components in
153 different schemas. So to check those reference-related constraints, related instance documents MUST
154 be validated using the same schema.

155 5 Interoperability Approach

156 We divide the universe of SML-IF documents into two disjoint subsets:

157 • A set that have all schema documents included, by value (smlif:data) and/or by reference
158 (smlif:locator), in the SML-IF instance; the "schema-complete set"

159 • All other SML-IF documents; the "schema-incomplete set"

160 It is necessary for a producer to declaratively distinguish between these two cases, since it is not always
161 possible to distinguish based on the content alone. For example, XML Schema allows xs:include's
162 schema location attribute's value to not resolve, although the value is required. This can be done by
163 introducing a "schemaComplete" attribute on the <smlif:definitions> element to indicate whether this SML-
164 IF instance includes all necessary definition documents.

165 When this attribute is specified with an actual value "true", then for the instance to be valid, its schema
166 definition documents and instance documents can only refer to either built-in components or components
167 from definition documents included in the instance.   "Built-in" components include:

168 • 4 xsi: attributes (defined by XML Schema)

169 • all schema built-in types (xs:anyType and simple types defined in XML Schema Part 2)

170 • sml:ref attribute declaration

171 • sml:uri element declaration

172 Remember, this is not trying to say that SML-IF document instances in the schema-incomplete set are
173 now invalid. It does say that SML-IF cannot guarantee interoperability for the schema-incomplete set.

174 # 6 Schema binding proposal

175 ## 6.1 An Example

176 (See the picture next page) Assume an SML model packaged in an IF document has 4 schema
177 documents: **xsd1-a** and **xsd1-b** have target namespace **ns1**, and **xsd2-v1** and **xsd2-v2** have target
178 namespace **ns2**, where **xsd2-v1** and **xsd2-v2** are conflicting versions of the same schema (same target
179 namespace). There are 4 instances: **doc1** uses **xsd1-a** and **xsd1-b**; **doc2-v1-a** and **doc2-v1-b** uses
180 **xsd2-v1**, and **doc2-v2** uses **xsd2-v2**. All **doc2-*** instances have SML references to **doc1**, and their
181 references have targetType constraints, pointing to a component in **ns1**.

182 To check targetType, **doc2-v1-a**, **doc2-v1-b** and **doc1** must be validated using the same schema (**xsd1-
183 a** + **xsd1-b** + **xsd2-v1**); similarly, **doc2-v2** and **doc1** must be validated using the schema from **xsd1-a** +
184 **xsd1-b** + **xsd2-v2**. More concretely, in the following picture, instances in the red rectangle are validated
185 using the schema built from schema documents in the red oval; and instances in the blue rectangle are
186 validated using the schema built from the blue oval.

187 Note that **doc1** is validated twice using 2 different schemas. **doc1** may also be validated against only
188 xsd1; this is up to the model author to specify.

189

190
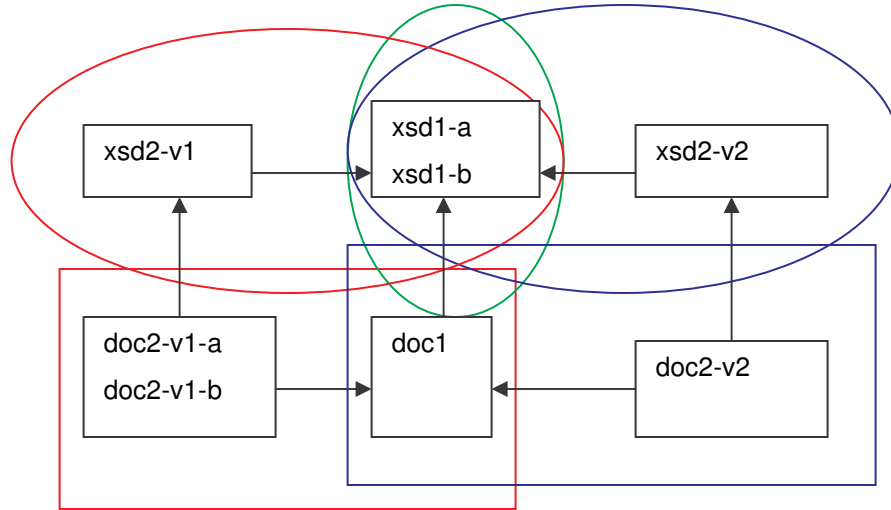
6.2 Solution to the Example

```
192      <schemaBindings>
193        <!-- Each "schemaBinding" element corresponds to a schema and model
194             instance documents that are assessed against this schema -->
195        <schemaBinding>
196          <!-- all "namespaceBinding" children together build the schema -->
197          <namespaceBinding namespace="ns1" aliases="xsd1-a xsd1-b"/>
198          <namespaceBinding namespace="ns2" aliases="xsd2-v1"/>
199          <!-- list all applicable instances; same as for rule bindings -->
200          <documentAlias>doc1</documentAlias>
201          <documentAlias>doc2-v1-a</documentAlias>
202          <documentAlias>doc2-v1-b</documentAlias>
203        </schemaBinding>
204        <schemaBinding>
205          <namespaceBinding namespace="ns1" aliases="xsd1-a xsd1-b"/>
206          <namespaceBinding namespace="ns2" aliases="xsd2-v2"/>
207          <documentAlias>doc1</documentAlias>
208          <documentAlias>doc2-v2</documentAlias>
209        </schemaBinding>
210      </schemaBindings>
211      <definitions schemaComplete="true">
212        <!-- schema documents for xsd1-a, xsd1-b, xsd2-v1, xsd2-v2 -->
213      </definitions>
```

214

6.3 Formal Proposal

215

216 1. Change the IF document structure to add the following (new content highlighted):

```
217     <model>
218       ...
219       <ruleBindings> ?
220         <ruleBinding> *
221           <documentAlias="xs:anyURI"/> ?
222           <ruleAlias="xs:anyURI"/>
223         </ruleBinding>
224       </ruleBindings>
225       <schemaBindings> ?
226         <schemaBinding> *
227           <namespaceBinding/> *   <!--
228                                       a single namespace name and
229                                       and list of schema document aliases -->
230           <documentAlias/> *      <!--
231                                       a list of instance document aliases -->
232         </schemaBinding>
233       </schemaBindings>
234       ...
235       <definitions schemaComplete="xs:boolean"> ?
236       ...
237     </model>
```

238 The details of the preceding XML syntax, e.g. whether the data is contained in attributes or elements,
239 is fully negotiable. The XML above simply captures enough to have the discussion that follows.

240 2. For every schema binding **SB** in the model, i.e. every "/model/schemaBindings/schemaBinding"
241 element (using XPATH notation):

242 2.1. Compose a schema using all documents specified under all **SB**'s <namespaceBinding> children

243 2.2. Whenever there is an <import> for a namespace **N**

244 2.2.1. If there is a <namespaceBinding> child of **SB** whose "namespace" matches **N**, then
245 components from schema documents listed in the corresponding "aliases" are used. As
246 with rule bindings, URI prefixing is used for matching schema document aliases.

247 Note: at most one <namespaceBinding> is allowed per namespace **N** within a given **SB**. If
248 more than one namespace binding exists for the namespace as part of a single schema
249 binding, the SML-IF instance is in error.

250 Note: if the set of aliases for namespace **N** is empty, the namespace has no schema
251 documents defining it in the schema binding.

252 2.2.2. Otherwise if there are schema documents in the IF whose targetNamespace is **N**, then
253 components from all those schema documents are used

254 2.2.3. Otherwise

255 2.2.3.1. If a schema-complete document (/model/definitions/@schemaComplete=true) is
256 being processed, then no component from N (other than built-ins) is included in the
257 schema being composed

258 2.2.3.2. Otherwise, it is implementation-defined whether the processor tries to retrieve
259 components for N from outside the SML-IF instance

260 2.3. Whenever there is an <include> or <redefine>, the schemaLocation is used to match aliases of
261 schema documents, as with base SML-IF.

262 2.3.1. If there is a schema document in the IF matching that alias, then that document is used

263   2.3.2.Otherwise

264   2.3.2.1. If it's a schema-complete set, then the <include> or <redefine> is unresolved (which
265   is allowed by XML Schema validity assessment rules)

266   2.3.2.2. Otherwise, it's implementation-defined whether it tries to resolve <include> or
267   <redefine> to schema documents outside the IF

268   2.4. The list of <documentAlias> documents are assessed against this *same* schema. targetXXX
269   and identity constraints can now be checked. Similar to <documentAlias> under <ruleBinding>
270   elements, each <documentAlias> can refer to multiple documents via URI prefixing.

271 3.   Compose a schema using *all* schema documents included in the IF, then use this schema to assess
272   those instance documents that are not included in any <schemaBinding>.

273 Note: in the common case where match-all namespace matching is the desired result, this is achieved by
274 omitting <schemaBindings>, i.e. without introducing any additional complexity into the SML-IF instance.

275 Note: one implication of this formulation is that the Schema document exchange requirement of section
276 3.5 is supported. This would be done by explicitly binding /model/instances/* to a schema binding that
277 excludes the exchange-only schemas. The model instance documents may still contain information items
278 from namespace(s) in the exchange-only schemas, however those schema documents would not be
279 used to assess schema validity of the model instance documents.

6.4 Proposal Analysis

281 •   Great synergy with <ruleBindings>

282   o   It works in a way very similar to Schematron rules. You associate a schema (built from a
283   set of schema documents) with a set of instance documents

284 •   Handles all the requirements

285   o   Supports schema composition: chameleon included documents is supported by removing
286   them from the corresponding <namespaceBinding> (whose "namespace" attribute is
287   absent)

288   o   Supports schema versioning: multiple versions can be specified in different
289   <schemaBinding> elements

290   o   Deterministic: the association between instances and schemas is deterministic

291   o   Full schema support: <include/redefine/import> are all supported

292   o   Schema document exchange: similar to chameleon included documents, exchange-only
293   documents can also be omitted from the corresponding <namespaceBinding>

294 •   Meets all the constraints

295   o   Supports access to schema documents outside of SML-IF: when schemaComplete=false,
296   processors are allowed to use external schema documents

297   o   Ignorable schema locations: all xsi:schemaLocation attributes can be ignored

298   o   Includes definition and instance documents as-is: no need to modify any included
299   document; document aliases are used.

300   o   Lazy schema assembly: the schema is known up-front; no need to handle lazy assembly

301   o   Supports reference constraints: instances specified under the same <schemaBinding>
302   use the same schema, so reference constraints can be checked.

303 •   Simple to understand

304 •   This has may Note that the "trivial case" is also handled by bullet 2. That is, there is no
305   <schemaBinding> and all instance documents are assessed against the same schema.

306 7 Implementation Cost

307 We have to assume that all existing schema processors are capable of handling the "namespace
308 matching" approach. That is, they can compose a schema from a list of schema documents.

309 This approach should be straightforward to handle. All the SML processor needs to do is to compute a
310 list of schema documents based on schema documents mentioned in <schemaBinding> and give that list
311 to the schema processor.

312 The "Explicit Binding" approach from earlier iterations of this proposal had what we believe is equivalent
313 function, but was eliminated because it also had greater complexity (four levels of binding to sift through,
314 instead of the two used here, i.e. schema binding and match-all namespace matching).

315 8 Acknowledgement

316 John Arwe, Bassam Tabarra, Harm Sluiman, and Pratul Dublish all provided useful input into the
317 formulation of this document. This does not imply their endorsement of the proposal.