Schema Versioning

Author:	Marko Alder
Report version:	1.0
GitHub issue:	–
Date:	October 15, 2019

Contents

Nomenclature **iii**

1 Context analysis **1**
1.1 Current situation 1
1.2 Aim of the proposal 1

2 Schema versioning **3**
2.1 CPACS schema versioning 3
2.1.1 Relevance of the `cpacsVersion` element 3
2.1.2 Three approaches to version the XML schema 4
2.2 Tool schema versioning 7

3 Conclusion **9**

4 Next steps **11**
4.1 Discussion of this proposal 11
4.2 Release of the results 11

Nomenclature

Abbreviations

CPACS	Common Parametric Aircraft Configuration Schema

1 Context analysis

1.1 Current situation

Currently CPACS (3.1) does not follow a **proper schema versioning strategy**.

The `header` element provides a `cpacsVersion` element. Humans can use this information to quickly indicate to which CPACS schema an XML instance file belongs to. However, data sets could be **compatible with multiple CPACS schema versions**. Thus, we must question the meaningfulness of this restriction.

1.2 Aim of the proposal

A consistent approach of schema versioning will be discussed.

2 Schema versioning

2.1 CPACS schema versioning

Currently the CPACS XML-schema is published without an official versioning strategy. In CPACS 3.1 the `cpacsVersion` element had been restricted to the float value *3.1*. Before discussing various approaches the relevance of this element needs to be discussed.

2.1.1 Relevance of the `cpacsVersion` element

From a users perspective it is quite convenient to parse for the `cpacsVersion` element to get an idea which XSD schema corresponds to a certain XML file. However, from a software perspective data should not be restricted to only one schema.

For example, if the definition of a rib positioning is modified, a dataset without internal structure must still be valid to the new schema version. This flexibility is required to ease the development of CPACS.

Another example becomes obvious when using the software library TiGL. A simple aircraft might be composed identically in CPACS 2.x and 3. However, TiGL 3 does not import the CPACS 2.x dataset if the `cpacsVersion` is not set to 3. However, the data might be CPACS 3.0 valid as well. TiGL has an internal schema file (should be identical to the latest CPACS release) and checks if the data is valid to that schema. Thus, the `cpacsVersion` is superfluous. On the other hand, it remains a good failure indicator for humans because the information that data is not compatible with schema version TiGL expects leads to an insight like: “Ah, to solve my error I need to look into the documentation of CPACS 3.x”.

A future solution should be that each software connecting to CPACS filters and checks the required data for validity with the CPACS schema it was developed for. If a software can handle multiple CPACS versions this could be an iterative process. The dataset must not pretend to correspond with a certain CPACS version - either it is valid or not.

The above discussion highlights that the information about the schema version (in the XML instance) only serves the traceability of data, but coupling the instance with a specific schema version is disadvantageous.

2.1.2 Three approaches to version the XML schema

There are basically three official approaches to version a schema file. These are summarized in `http://www.xfront.com/Versioning.pdf`. Some argumentations are copied from this document.

Option 1: Internal version attribute

The first option provides information about the schema version only in the schema file itself. In this variant, the actual data record does not contain any information about the underlying schema version:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            ...
            targetNamespace="http://www.cpacs.de"
            xmlns="http://www.cpacs.de"
            elementFormDefault="qualified"
            version="3.2">
```

Table 2.1: Pros and cons of option 1.

pro	con
Easy to implement	The validator ignores the version attribute.
XML file would not have to change if it remains valid with a new version of the schema (see Sec. 2.1.1)	No traceability of data in the XML instance.
The schema provides information which informs an application that has changed. An application could parse the version attribute, recognize that this is a new version of a schema take appropriate action if necessary.	
A user who opens the schema file has an indication to which CPACS version it belongs to.	

Option 2a: Required version attribute on the root element

The schema version can be a required attribute of the CPACS root element. This is basically equivalent to the current approach where the `cpacsVersion` element is required in the `header`, but follows common practices to version datasets.

```
<xsd:complexType name="cpacsType">
  <xsd:all>
    <xsd:element name="header" type="headerType"/>
    <xsd:element minOccurs="0" name="vehicles" type="vehiclesType"/>
    ...
  </xsd:all>
  <xsd:attribute name="schemaVersion" type="xsd:decimal" use="required"
     fixed="3.2"/>
</xsd:complexType>
```

```
<?xml version="1.0" encoding="UTF-8"?>
  <cpacs xmlns="http://www.cpacs.de"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:ext="http://www.cpacs.de/liftingLine"
       xsi:schemaLocation="http://www.cpacs.de cpacs_schema.xsd"
       schemaVersion="3.2">
<header>
```

Table 2.2: Pros and cons of option 2a.

pro	con
The `schemaVersion` attribute is an enforceable contraint.	The `schemaVersion` value must match exactly. This does not allow a XML instance to be valid with multiple schema versions.

Option 2b: Optional version attribute on the root element

This option is similar to the approach discussed above with the exception that the `schemaVersion` attribute is not required. Thus, a data set can be valid with multiple schemas until the attribute is set.

```
<xsd:complexType name="cpacsType">
<xsd:all>
<xsd:element name="header" type="headerType"/>
<xsd:element minOccurs="0" name="vehicles" type="vehiclesType"/>
...
</xsd:all>
<xsd:attribute name="schemaVersion" type="xsd:decimal" fixed="3.2"/>
</xsd:complexType>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<cpacs xmlns="http://www.cpacs.de"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:ext="http://www.cpacs.de/liftingLine"
 xsi:schemaLocation="http://www.cpacs.de cpacs_schema.xsd"
 schemaVersion="3.2">
 <header>

```

Table 2.3: Pros and cons of option 2b.

pro	con
The `schemaVersion` attribute will be checked if it is set.	The `schemaVersion` value must match exactly. This does not allow a XML instance to be valid with multiple schema versions.

Option 3: Change of the schema's target namespace

The namespace is identified by an arbitrary URI[1]. This namespace could include the schema version:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      ...
      targetNamespace="http://www.cpacs.de/v3.2"
      xmlns="http://www.cpacs.de/v3.2"
      elementFormDefault="qualified">
```

Table 2.4: Pros and cons of option 3.

pro	con
Requires action to assure that there are no compatibility problems with the new schema.	Instance documents depend on a specific schema version.

We think that compatibility should not be assured by the namespace name because it is very uncommon in software design. E.g., lets look at the example of a package manager handling a python environment. The user should say "install numpy" letting the package manager decide which version is compatible to the pre-installed modules.

[1] `https://www.w3.org/TR/xml-names`

Option 4: Change of the name/location of the schema

We make the scheme available online on the CPACS homepage[2]. This could support the traceability of the instantiated data:

```
<?xml version="1.0" encoding="UTF-8"?>
<cpacs xmlns="http://www.cpacs.de"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.cpacs.de http://www.cpacs.de/schema
    /v3_1_0/cpacs_schema.xsd">
```

Table 2.5: Pros and cons of option 4.

pro	con
The instance document implicitly contains information about the schema version, but it does not have to (if some local files are linked)	The complete data depends again on a single schema version. All instance documents need to be changed even if the cchange of the schema would not impact the instance.
	The schemaLocation attribute in the instance document is optional and is not authoritative even if it is present. It is a hint to help the processor to locate the schema. Therefore, relying on this attribute is not a good practice (with the current reading of the specification).

2.2 Tool schema versioning

The introduction of tool-specific namespaces is discussed in GitHub issue #484[3]. Since the corresponding tool-specif schema file is in the responsibility of the tool vendor everyone is free to choose its own versioning strategy. Only the corresponding tool depends on that data and the degree of coupling the schema version and the tool version depends on the developer.

However, the tool-version and tool-specific schema version is given as optional attribute to support traceability of data.

[2] http://www.cpacs.de/schema/v3_1_0/cpacs_schema.xsd
[3] https://github.com/DLR-SL/CPACS/issues/484

3 Conclusion

Section 2.1.1 concludes that the information about the schema version (in the XML instance) only serves the traceability of data, but coupling the instance with a specific schema version is disadvantageous.

We therefore propose the following: Option 1 (adding a version attribute to the schema header) supports a proper versioning of the schema file itself. An element like `cpacsVersion` linking the complete dataset with a single namespace version will be removed in future version. However, since traceability of data is important for the evaluation of results we must find a proper strategy to trace specific datasets. This can be done by extending the `baseTypes` with additional attributes (e.g., referring to the schema version, software version, time and date of data creation, ...).

As long as we develop such data traceability strategy the `cpacsVersion` element will be changed to be an optional element. Thus, we can decide if we need the information in a specific work-flow or not. Option 2b would fulfill the same requirement. However, if the strategy is to get rid of the dependency to a single schema version it seems unnecessary to introduce this approach and remove it again in future version.

If it is required in work-flow that the schema-version must be fixed for a certain reason, option 2b could be introduced instead. However, a recent survey indicates that this is not the case.

Option 4 is implicitly given because the schema files can be reference online. But providing the `schemaLocation` attribute is optional and each tool can reference its filtered dataset to a specific schema version.

4 Next steps

4.1 Discussion of this proposal

The CPACS community is informed about the proposed changes by this document and comments on GitHub or e-Mail[1] to add additional aspects which must be considered.

4.2 Release of the results

If we follow this proposal the `cpacsVersion` element will changed to be optional and option 1 will be introduced in the next CPACS release.

[1] cpacs@dlr.de