*sub*check Article MarkupUK London

Andreas Tai, Institut für Rundfunktechnik (IRT)

Michael Seiferle, BaseX GmbH <ms@basex.org>

Abstract

The exchange of data in XML goes hand in hand with its validation. Mature technologies for this task exist. But often more is needed than just the technical application of XML Schema, Schematron or Relax NG. A user-friendly solution requires an abstraction from the XML foundation. Initiation and result of a validating process needs to be put into the context of the end users domain and be a guidance to solve conformance problems.

This paper shows how Schematron, XSLT, XQuery and JavaScript have been glued together to address these requirements. Although the subcheck framework (http://subcheck.io [http://subcheck.io/]) was implemented specifically for the Timed Text Markup Language (TTML) the approach can be used by any other XML vocabulary.

Table of Contents

Intro

Do you share this problem? You have a well-documented XML structure and an XML Schema to test conformance, but:

- People do not use the XML Schema because they do not know it exists.
- People know the XML Schema exists, but they do not know how to apply it.
- People apply the XML Schema, but they do not know how to interpret the error messages.

If so: you are in the same situation as we were in 2012.

TTML and its Profiles

The W3C Timed Text Markup Language (TTML) [TTML2] is an XML vocabulary for video subtitles and captions.

Like many other XML vocabularies, TTML works as a baseline and covers many known use cases. One use case is the exchange of subtitle files, another use case is the rendering of subtitles in a video player. That's why it comes to no surprise that in operation not the complete TTML vocabulary is used, but only subsets of the Markup Language.

Examples of these subsets are EBU-TT-D [EBUTTD] defined by the European Broadcast Union (EBU) and the widely adopted TTML Profiles for Internet Media Subtitles and Captions (IMSC) [IMSC11] defined by the Timed Text Working Group (TTWG) [1]of the World Wide Web Consortium (W3C), that also published TTML.

Validation of TTML Profiles

Especially in the introduction phase of a new standard, it is important to make sure that the APIs of different software systems use the standards in the same way and conform to the rules of the specifications.

Once a workflow is established, the exchanged documents need to conform to the established APIs.

Take for example the following real-world use case scenario: a freelancer creates subtitles for a media service company which in turn got a contract from a streaming company to deliver subtitles for a complete season of a video series.

The media service company and the streaming company agree on IMSC as the technical baseline for subtitle XML files, but they may also apply their own set of "house style rules". So at the gateways of the service company and the streaming provider, overlapping but also separate checks are executed.

The Institute für Rundfunktechnik (IRT) [2]was well aware of this and other scenarios such as the "in-house" exchange of subtitle files and aimed for a better support for them.

In 2013, the IRT had already been active in TTML standardization, especially in the definition of the EBU-TT profiles.[3] But although W3C XML Schema (XSD) files existed, non-conformant documents were still produced and distributed.

This was due to a lack of knowledge about the existence of the XSD itself, its application or about the meaning of the implied rules and resulted in the new goal to build a more user-friendly validation alternative. The main inspiration came from the W3C HTML validation service.

In the process of the installation of their own service, the IRT went through different stages:

- A master thesis on the topic was written by Barbara Fichte (supervised by Prof. Dr. Anne Brüggemann-Klein) and a first prototype was implemented.
- A cooperation with the company BaseX GmbH was started.
- Full working solutions were implemented.
- Two different products (*sub*check and IMF Analyser) were launched to integrate the solutions.

In the following, we describe the implementation of the *sub*check service by BaseX GmbH and the IRT.

Validation Requirements

One important requirement was to validate a TTML XML document against different sets of conformance rules without duplicating the implementation effort, another that the end user should only get one error message when the same rule of different specifications was broken. On top of that, the implementation of the software architecture was also guided by a number of non-functional requirements.

Team Expertise

The choice of technologies had to reflect the expertise available in the technical teams. For example, the IRT team worked intensively with XSLT, Schematron and W3C XML Schema but less with XQuery or Relax NG.

Deployment of Technologies

[1]https://www.w3.org/AudioVideo/TT/
[2]The Institut für Rundfunktechnik GmbH (IRT) (Institute for Broadcasting Technology Ltd.) is the research centre of the German broadcasters, Austria's broadcaster and the Swiss public broadcaster.
[3]see https://tech.ebu.ch/groups/subtitling

The technologies used needed to be widely deployed because it had to be possible to integrate the solution into different system environments.

All XML

We believed and still believe in the power of XML technologies. The cooperation of communities responsible for different XML technologies resulted in well-aligned XML technologies like XPath, XSLT, XQuery, Schematron and XPROC. We therefore wanted to use XML technologies wherever possible.

Separation of Concerns

We favoured a design approach with a framework of loosely coupled components that communicate over an agreed API. This way, different stakeholder groups can work on different parts of the framework without interfering with the work of other stakeholders.

We identified three different stakeholder groups:

- The domain stakeholders who know about conformance rules.
- The developers that implement the rules using validation technologies.
- The product team that technically designs and implements the end product.

End user Requirements

Existing validation strategies often come with very technical result messages, but users are often not technical. Even if they are, they may not be familiar with an XML schema technology or XML at all. Therefore, the validation should give information in an understandable way. The information should highlight not only what is wrong but also WHY something is wrong and how to correct it. It should enable the user to trace back the error to the documentation of a rule in the specification.

Product View

From the commercial side, some components needed to be integrated separately into different products. Different developer teams needed to work independently from each other.

Implementation Approach

Master Thesis

Barbara Fichte explored the topic in depth in her well written master thesis "Strategies for User-Oriented Conformance Testing of XML Documents".[MAFICHTE] She worked out the requirements, evaluated different schema languages and described how she implemented a first prototype.

She looked at the schema languages W3C XML Schema, Schematron, Relax NG and NVDL. The conclusion was that Schematron was the best fit for the purpose. One major aspect was the expertise in the team. But most importantly Schematron makes it easy to integrate additional information into the validation.

The other Schema technology that was favoured was W3C XML Schema. It is widely implemented and works well with grammar-based constraints. But W3C XML Schema came with a number of limitations. The most important one was that for some types of rules XML Schema 1.1 was needed and at the time of writing of the master thesis only limited support by free or open-source XSD 1.1 schema parsers was available.

As Schematron was such a good fit, Barbara Fichte proposed and implemented an approach where all constraints could be implemented with Schematron. She used Schematron’s abstract patterns to implement grammar constraints that would usually be implemented with W3C Schema. One example of these constraints is the order in which elements may appear under a parent element.

In the later implementation of *sub*check, the main schema technology used was indeed Schematron. However, we did not completely remove W3C XML Schema from the validation process. The costs

to re-implement already existing grammar constraints in Schematron was too high compared to the benefit. There also existed some XML Schema of TTML profiles published by standard bodies. The use of these schemas enabled *sub*check validation to align better with the validation approaches of these organizations.

Application Implementation

In the final implementation, we separated three different steps:

- The documentation of the rules in a constraints XML document.
- The implementation of the rules in Schematron.
- The creation of a report that could be used by an end user product.

As the approach we took can be applied to any XML vocabulary in many scenarios, we will present a simplified use case to visualize the key points.

The scenario is to test cabin bag weight against the allowance of the two assumed airlines Aeto and Örn.

In the appendix, different examples show how the approach can be applied to TTML vocabularies.

Rules Documentation

The base of the *sub*check framework is the documentation of the rules in a separate XML file (constraints XML).

The implementation assumption is that more than one specification needs be tested. These specifications need to be defined first. To keep the XML samples in the paper short we later use only one specification.

```
<Specifications>
    <Specification ID="ID-Aeto-Conditions">
        <Name>Conditions Aeto</Name>
        <Acronym>C-Aeto</Acronym>
        <Version>1.0</Version>
    </Specification>
    <Specification ID="ID-Oern-Conditions">
        <Name>Conditions Örn</Name>
        <Acronym>C-Örn</Acronym>
        <Version>1.0</Version>
    </Specification>
</Specifications>
```

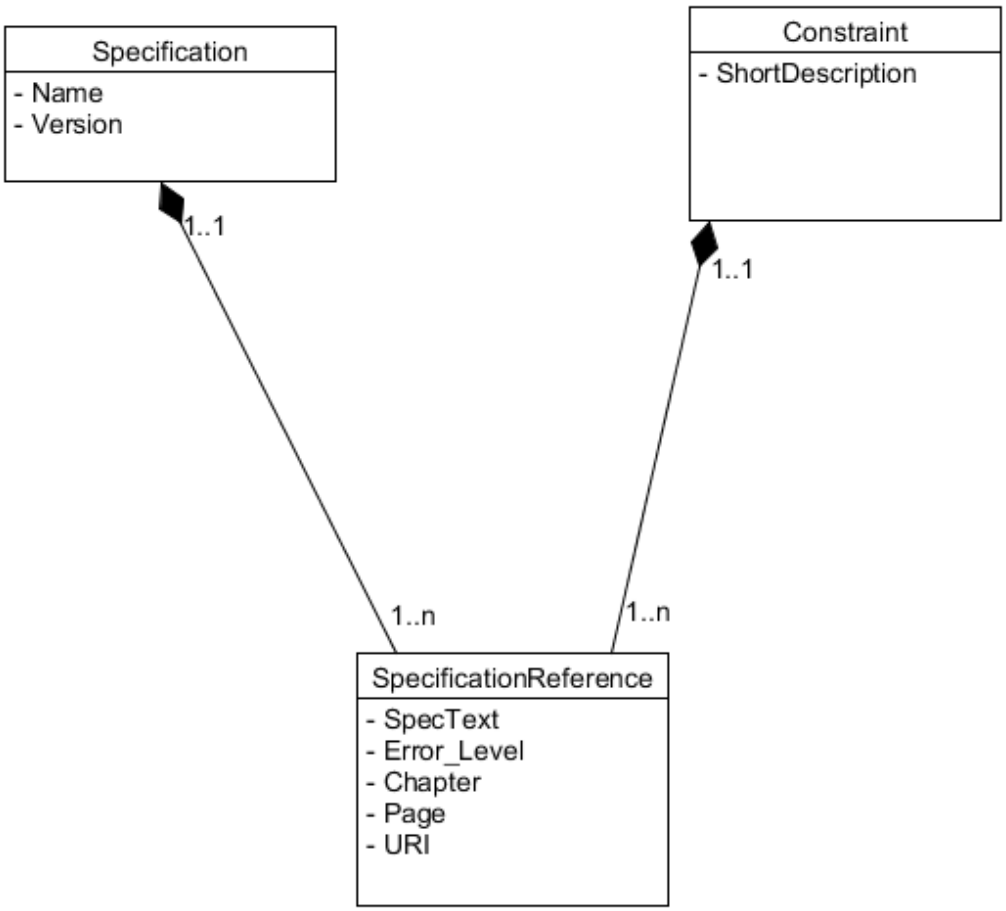
After that, the constraints can be documented with the necessary level of detail and linked back to the specification.

```
<Constraint ID="c1">
    <ShortName>
      Cabin Bag Max. Weight 8kg
    </ShortName>
    <SpecifiedBy ID="ID-cabinbag-8kg">
        <SpecificationReference>
          ID-Aeto-Conditions
         </SpecificationReference>
         <Error_Level>ERROR</Error_Level>
    </SpecifiedBy>
    <ShortDescription>
      Value should be equal or less than 8.
    </ShortDescription>
```

```
    <ShortDescriptionUser>
      Cabin bag should not have more than 8kg weight.
    </ShortDescriptionUser>
</Constraint>
```

The following UML class diagram models the relationship between constraints and specifications.

Figure 1. UML Diagram Constraint - Specification



All information documented in the constraints XML should be specific to the constraint and not its implementation.

One constraint can be part of different specifications and each of these specifications may put the constraint in a different context. While one specification may specify the constraint as a mandatory requirement, another specification may just regard it as recommendation. The first context would result in an error and the second context would result in a warning.

Schematron Schema File

The rule is implemented in Schematron as defined by the Schematron standard [ISOSCHEMA]. In addition, the `@see` attribute is used for linking the Schematron rule back to the documented constraint.

```
<sch:rule context="cabin-bag/weight">
    <sch:let
      name="bag-weight"
      value="xs:float(.)"/>
    <sch:let
```

```
      name="passenger-name"
      value="../../name"/>
    <sch:assert
      test="$bag-weight le 8"
      see="http://www.cabin-bag.info/c1"
      diagnostics="diag-weight-8">
        The weight of cabin luggage is 8kg
        or less.
    </sch:assert>
</sch:rule>
```

More context information is given by using Schematron's diagnostic element.

```
<sch:diagnostic id="diag-weight-8">
  The cabin luggage of
  <sch:value-of select="$passenger-name"/>
  exceeded the maximum weight allowance by
  <sch:value-of select="$bag-weight -8"/>kg.
  Pack lighter!
</sch:diagnostic>
```

Compiled Schematron

The Schematron Schema is compiled into an XSLT using a customized version of the Schematron skeleton implementation [4]. The sekeleton implementation needed to be adjusted for two reasons:

1. We also wanted to use attributes as rule context.[5]
2. We wanted to provide a more human readable version of the location where the error occurs.

Given the the following XML file…

```
<passenger>
    <name>Jane Grant</name>
    <cabin-bag>
        <weight>11</weight>
    </cabin-bag>
 </passenger>
```

…the target output of the compiled Schematron XSLT is an XML document that follows the structure of the Schematron Validation Report Language. A failed Schematron assert would result in the following SVRL:

```
<svrl:failed-assert
  test="$bag-weight le 8"
  id="assert-c1-1"
  see="http://www.cabin-bag.info/c1"
  location="/passenger[1]/cabin-bag[1]/weight[1]"
  subcheck:alternativeLocation="/passenger/cabin-bag/weight">
   <svrl:text>
     The weight of cabin luggage is 8kg or less.
   </svrl:text>
   <svrl:diagnostic-reference
      diagnostic="diag-weight-8">
```

[4] see https://github.com/Schematron/schematron

[5] see also the discussion on issues https://github.com/Schematron/schematron/issues/44 and https://github.com/Schematron/schematron/issues/29 and pull request https://github.com/Schematron/schematron/pull/41

```
        The cabin luggage of Jane Grant exceeded
        the maximum weight allowance by 3kg.
        Pack lighter!
    </svrl:diagnostic-reference>
</svrl:failed-assert>
```

The reference to the documentation of the constraint in the constraint.xml is kept in a `@see` attribute.

Generation of the Report

The SVRL is taken as input for the transformation into an XML report. This report is provided to the end user product that integrates the validation. The XML structure of the report is the agreed API between validation engine and post-processing system.

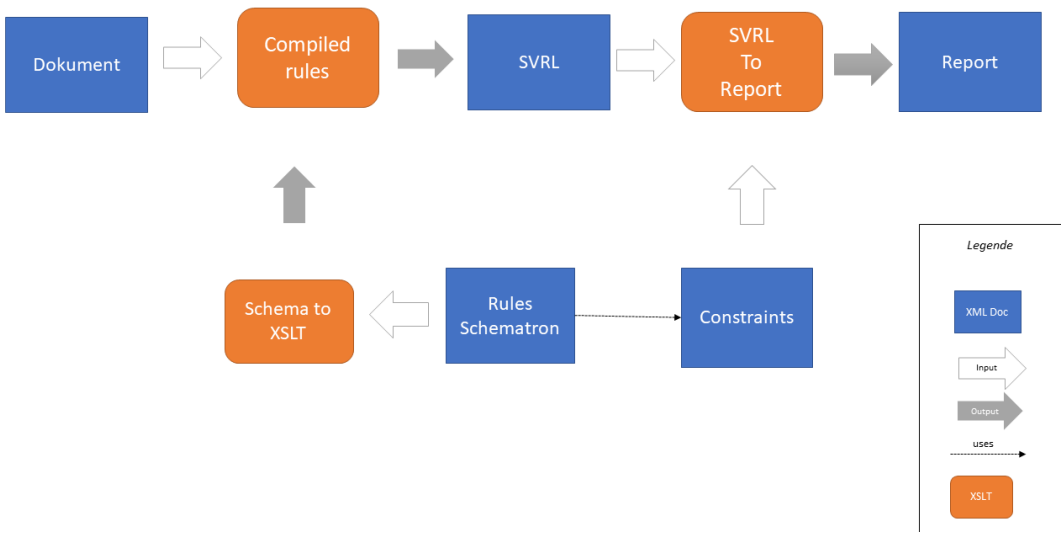
The output is grouped by constraint in the report.

```
<errorCategory>
    <constraintID>
      c1
    </constraintID>
    <title>
      Cabin Bag Max. Weight 8kg
    </title>
    <shortUserDesc>
      Cabin bag should not have more
      than 8kg weight.
    </shortUserDesc>
    <longUserDesc/>
    <specs>
        <spec>
            <name>
              Conditons Aeto, Version 1.0
             </name>
            <nameAcronym>
              C-Aeto<
            /nameAcronym>
            <errorLevel>ERROR</errorLevel>
        </spec>
    </specs>
    <errors>
        <error>
            <messages>
                <messageMain>
                  Assertion: The weight of cabin
                  luggage is 8kg or less.
                  Error Information: The cabin
                  luggage of Jane Grant exceeded
                  the maximum weight allowance
                  by 3kg. Pack lighter!
                </messageMain>
                <messageAssertion>
                  The weight of cabin luggage is
                  8kg or less.
                </messageAssertion>
                <messageDiagnosticsAll>
                  The cabin luggage of Jane Grant
                  exceeded the maximum weight
                  allowance by 3kg. Pack lighter!
```

```
                </messageDiagnosticsAll>
            </messages>
            <locations>
                <location
                  locationType="resolvableXPATH">
                  /passenger[1]/cabin-bag[1]/weight[1]
                </location>
                <location
                  locationType="humanXPATH">
                  /passenger/cabin-bag/weight
                </location>
            </locations>
        </error>
    </errors>
</errorCategory>
```

Overview: The Transformation Chain

Figure 2. Transformation in the Validation Chain



Interfaces: Machine-to-Machine and the End-User

*sub*check's core interfaces are based on RESTXQ and are implemented in the BaseX XML Server as a pure XQuery application.

The service contains a single endpoint `/validate` that orchestrates the validation engine and returns the resulting report. Files that are uploaded by users will not be persisted by the engine, they are parsed and validated in memory only and returned to the caller right away. The report will be returned in its XML form by default, but other formats are possible.

The XQuery implementation receives an XML file — it returns an error on non-XML files — and runs the XSL transformation for the compiled rules. The result of this process is then passed on to another stylesheet that transforms it to the reporting format and if necessary conducts conversions to other formats such as JSON or text.

In practice, a request looks like the following:

```
$ cat luggage.xml | http POST http://subecheck/validate
#####################################################
> Content-Type: application/xml; charset=UTF-8


  <report>
    <errors>
    <constraintID>c1</constraintID>
    <title>
     Cabin Bag Max. Weight 8kg
    </title>
    <shortUserDesc>
      Cabin bag should not have more than 8kg weight.
    </shortUserDesc>
    ...
  </report>
```

Via REST, users can also request a report in JSON format by specifying a different `Accept`-Header:

```
$ cat luggage.xml | http POST http://subecheck/validate Accept:application/json
#############################################################################
> Content-Type: application/json; charset=UTF-8


{
  "filename": "luggage.xml",
  "filesize": "2 KB",
  "report": [{
    "constraintID" : "c1",
    "title"        : "Cabin Bag Max. Weight 8kg",
    "shortUserDesc": "Cabin bag should not have more than 8kg weight.",
    "longUserDesc" : "Assertion: The weight of cabin […]",
    "specs"        : [{
      "errorLevel" : "ERROR",
      "name"       : "Conditons Aeto, Version 1.0",
      "nameAcronym": "C-Aeto",
      "section"    : "[…]",
      "text"       : "[…]",
      "uri"        : "https://c-aeto/spec"
    }],
```

While *sub*check was designed with arbitrary frontends in mind — i.e. it can be easily integrated into existing workflows thanks to its almost universally accessible REST interface — we decided to implement a visual interface as a proof-of-concept that allows users to interactively explore and assess their validation results.

This interface heavily builds on the `JSON`-serialization of the reports and is implemented as a Single-Page-Application made up of `Vue.js`-components, that allow the user to potentially browse and filter hundreds of validation messages.

Using the *sub*check Application

In order to validate an XML file, users may initiate validation by dragging their file onto the browser window. Once the file is dropped, it is sent to the server and the validation pipeline will start. Once the pipeline has finished, usually in under a second, the result list will appear.

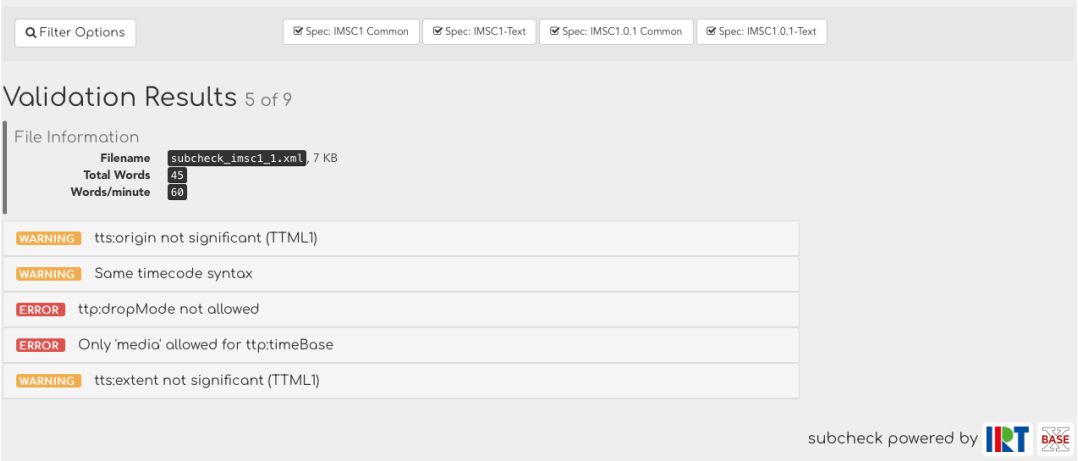
The following screenshots are taken from the examples section at https://subcheck.ioand are the results of validating a real subtitle.

The Report View

The report view allows the display of general metadata, such as word counts or other metrics, and allows users to gather insight into the validation results and filter them according to their needs.

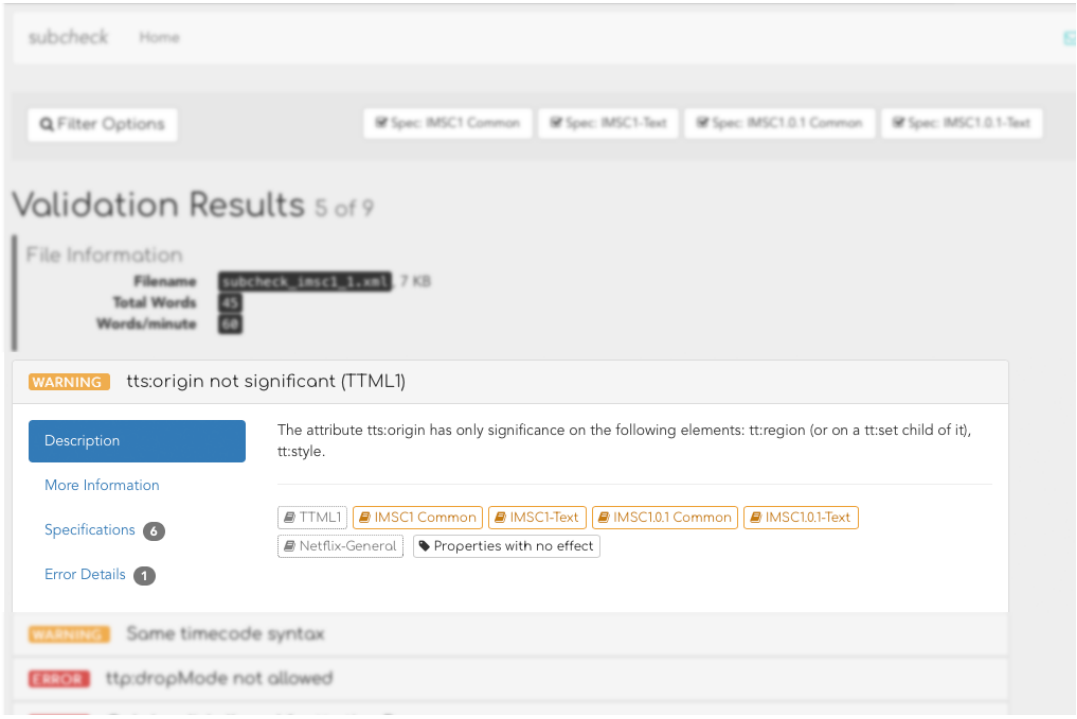
This view is built completely on the report result — so all information displayed is also available to other engines or systems.

Figure 3. The Report View



Inside the Report View, users may click on the messages to display more in-depth information.

Figure 4. Filtering



Tabs Overview

Description Tab

The detail view starts with a general error description to add more technical context for the user. When working with different profiles within the same standard, *sub*check allows adding `Tags` to given rules that will be shown along with the general description, so users

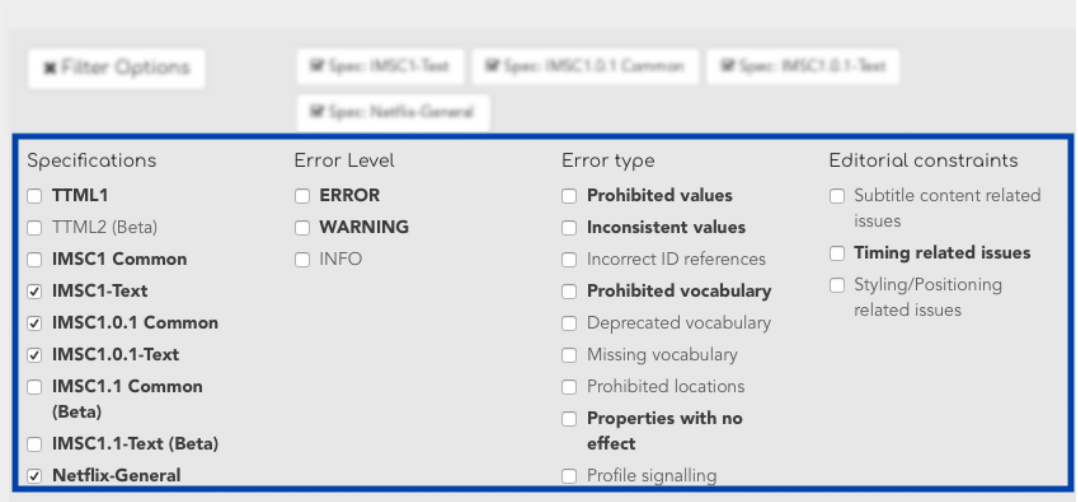
	can quickly assess whether this rule's `ErrorLevel` is set differently in other profiles.
More Information Tab	In addition to technical information, this tab will provide more general or editorially useful information and hints on why that message appeared.
Specifications Tab	The Specifications tab points to the actual specifications this error violates. Usually entries contain a link that enables users to read the original specification for further information.
Error Details Tab	This tab contains an `XPath`-Expression that can be copied to any capable editor or engine to pinpoint the exact error location. An HTML preview of the error is also provided, so the user gets some document context.

Adaptive Filtering

A *sub*check application can be used to validate documents against arbitrary schemas and allows users to provide user-defined tags that apply to a given validation message.

The screenshot below shows filtering based on a real-world subtitle validation framework:

Figure 5. Filtering



The frontend automatically adapts itself to the validation results, in a way that allows users to explore their data with these facets. The screenshot above shows, highlighted in **bold**, what kinds of messages are contained within the result. Users may check or uncheck these categories to display or hide specific items in the report view.

For example, some users might only be interested in messages with a `WARNING`-Level, or maybe only warnings that apply to styling-related messages.

Conclusion

The separation of concerns was successfully implemented.

Users can write the rules. They can update the documentation without interfering with validation or product design.

Developers can implement the validation rules. They do not have to rely on the domain user (unless there is a new rule that is a breaking change). Although the main implementation is in Schematron, other schema languages can be used as well, as long as a reported error links back to the documented rule.

Product designers can rely on the fact that, on the one hand, the same API is triggered to submit the file and filter criteria and, on the other hand, to always get back an XML document in the same format.

Other Aspects and Perspective

Some interesting aspects could not be covered in this paper but should be mentioned shortly:

- *sub*check was also used to validate the binary subtitle file format EBU STL. To make this possible the binary STL file was translated first into an XML representation of the binary structure. After that, the resulting XML structure was validated using Schematron rules.
- Apart from the *sub*check service, the validation engine was also implemented in the workstation-based product IMF analyser. The tool is written in C# and C++.
- Especially the integration of the W3C XML Schema validation was a challenge. It needs more work and design changes to become a generic solution.

Bibliography

[EBUTTD] TECH3380 EBU-TT-D SUBTITLING DISTRIBUTION FORMAT. W3C https://tech.ebu.ch/publications/tech3380

[ISOSCHEMA] ISO/IEC 19757-3, Information technology — Document Schema Definition Languages (DSDL) - Part 3: Rule-based validation - Schematron. 15 January 2016, ISO/IEC http://standards.iso.org/ittf/PubliclyAvailableStandards/c055982_ISO_IEC_19757-3_2016.zip

[MAFICHTE] Barbara Fichte: Strategien zur benutzerorientierten Konformitätsprüfung von XML-Dokumenten. 15 June 2014, Technische Universität München

[IMSC11] Pierre Lemieux: TTML Profiles for Internet Media Subtitles and Captions 1.1. W3C https://www.w3.org/TR/ttml-imsc1.1/

[TTML2] Glenn AdamsCyril Concolato: Timed Text Markup Language 2 (TTML2). W3C https://www.w3.org/TR/ttml2/

A. Appendix A - Subcheck artifacts with TTML examples

Example of TTML specification listing in the constraints.xml:

```
<Specifications>
  <Specification ID="spec-imsc1.1">
         <Name>
            TTML Profiles for Internet Media
            Subtitles and Captions 1.1
          </Name>
         <Acronym>IMSC1.1 Common</Acronym>
         <Version>2018-11-08</Version>
  </Specification>
  <Specification ID="spec-ttml2">
         <Name>
            Timed Text Markup Language 2
            (TTML2) CR
         </Name>
         <Acronym>TTML2</Acronym>
         <Version>2018-11-08</Version>
  </Specification>
```

```
</Specifications>
```

Example of a documented IMSC rule:

```
<Constraint ID="d1e1321">
    <ShortName>
        Extent attribute presence on region
        element
   </ShortName>
    <SpecifiedBy>
        <SpecificationReference>
            spec-imsc1-text
        </SpecificationReference>
        <SpecText>
         [The feature] #extent-region [is]
         permitted...
        </SpecText>
        <Error_Level>ERROR</Error_Level>
        <Chapter>7.4</Chapter>
        <URI>
          https://www.w3.org/TR/ttml-imsc1/#features-and-extensions-1
        </URI>
    </SpecifiedBy>
    <SpecifiedBy>
        <SpecificationReference>
            spec-imsc1.0.1-text
        </SpecificationReference>
        <SpecText>...</SpecText>
        <Error_Level>ERROR</Error_Level>
        <Chapter>7.4</Chapter>
        <URI>
            https://www.w3.org/TR/ttml-imsc1.0.1/#features-and-extensions-1
        </URI>
    </SpecifiedBy>
    <SpecifiedBy>
        <SpecificationReference>
            spec-imsc1.1-text
        </SpecificationReference>
        <SpecText>...</SpecText>
        <Error_Level>ERROR</Error_Level>
        <Chapter>8.4.2</Chapter>
        <URI>
           https://www.w3.org/TR/ttml-imsc1.1/#extent-region
        </URI>
    </SpecifiedBy>
    <ShortDescription>
        tts:extent on all regions
    </ShortDescription>
    <ShortDescriptionUser>
        The extent attribute shall be present on all region
        elements.
    </ShortDescriptionUser>
</Constraint>
```

Example of rule implementation in Schematron:

```
<sch:pattern id="attributeRequirement">
    <sch:rule
     context="/tt:tt/tt:head/tt:layout/tt:region">
```

```
      <sch:assert
       diagnostics="elementId"
       see="http://www.irt.de/subcheck/constraints/d1e1321"
       id="assert-d1e1321-1">
        The attribute tts:extent is present.
      </sch:assert>
    </sch:rule>
</sch:pattern>
```

Example of an IMSC document with an error:

```
<tt xmlns="http://www.w3.org/ns/ttml"
    ttp:profile="http://www.w3.org/ns/ttml/profile/imsc1/text"
    xmlns:ttp="http://www.w3.org/ns/ttml#parameter"
    xmlns:tts="http://www.w3.org/ns/ttml#styling"
    xml:lang="en">
    <head>
        <layout>
            <region tts:origin="10% 80%"
             xml:id="bottom"/>
        </layout>
    </head>
    <body>
        <div>
            <p region="bottom" begin="0s" end="1s">
                Hello, I am Mork from Ork.
            </p>
        </div>
    </body>
</tt>
```

Example of the SVRL Output:

```
<svrl:failed-assert
   test="attribute::tts:extent"
   see="http://www.irt.de/subcheck/constraints/d1e1321"
   location="/*:tt[namespace-uri()='http://www.w3.org/ns/ttml'][1]/..."
   subcheck:alternativeLocation="/tt/head/layout/region">
    <svrl:text>
        The attribute tts:extent is present.
    </svrl:text>
    <svrl:diagnostic-reference
     diagnostic="elementId">
       The affected 'region' element has the ID 'bottom'.
    </svrl:diagnostic-reference>
</svrl:failed-assert>
```

Example of the report view output:

```
<errorCategory>
    <constraintID>d1e1321</constraintID>
    <title>
        Extent attribute presence on region element
    </title>
    <shortUserDesc>
        The extent attribute shall be present on all
        region elements.
    </shortUserDesc>
    <specs>
        <spec>
```

```
            <name>
                TTML Profiles for Internet Media
                Subtitles and Captions 1.0 (IMSC1)-
                Text Profile, Version 2016-04-21
            </name>
            <nameAcronym>IMSC1-Text</nameAcronym>
            <text>
                [The feature] #extent-region [is]
                permitted ...
            </text>
            <errorLevel>ERROR</errorLevel>
            <section>Chapter 7.4</section>
            <uri>
                https://www.w3.org/TR/ttml-imsc1...
            </uri>
        </spec>
        <spec>
            <name>
                TTML Profiles for Internet Media
                Subtitles and Captions 1.0.1 (IMSC1)
                - Text Profile, Version 2017-07-13
            </name>
            <nameAcronym>IMSC1.0.1-Text</nameAcronym>
            <text>
                [The feature] #extent-region [is]
                permitted...
            </text>
            <errorLevel>ERROR</errorLevel>
            <section>Chapter 7.4</section>
            <uri>
               https://www.w3.org/TR/ttml-imsc1.0.1/...
            </uri>
        </spec>
        <spec>
            <name>
                TTML Profiles for Internet Media
                Subtitles and Captions 1.1 -
                Text Profile,Version 2017-10-17
            </name>
            <nameAcronym>IMSC1.1-Text (Beta)</nameAcronym>
            <text>
                The tts:extent attribute SHALL be present
                on all region elements, where it SHALL use
                px units, percentage values, or root
                container relative units.
            </text>
            <errorLevel>ERROR</errorLevel>
            <section>Chapter 8.4.2</section>
            <uri>
                https://www.w3.org/TR/ttml-imsc1.1/...
            </uri>
        </spec>
    </specs>
    <errors>
        <error>
            <messages>
                <messageMain>
                   Assertion: The attribute tts:extent is
```

```
                    present.
                    Error Information:
                    The affected 'region' element has the ID
                   'bottom'.
                </messageMain>
                 <messageAssertion>
                    The attribute tts:extent is present.
                 </messageAssertion>
                 <messageDiagnosticsAll>
                     The affected 'region' element has the ID
                     'bottom'.
                 </messageDiagnosticsAll>
             </messages>
             <locations>
                 <location
                   locationType="resolvableXPATH">
                     /*:tt[namespace-uri()='http://www.w3.org/...
                 </location>
                 <location
                  locationType="humanXPATH">
                     /tt/head/layout/region
                 </location>
             </locations>
         </error>
     </errors>
</errorCategory>
```