EAD-ODD: A solution for project-specific EAD schemes

Laurent Romary, Charles Riondet

EAD-ODD: A solution for project-specific EAD schemes

Laurent Romary, Charles Riondet
Inria Paris, ALMAnaCH[1]

This article tackles the issue of integrating heterogeneous archival sources in one single data repository, namely the European Holocaust Research Infrastructure (EHRI) portal, whose aim is to support Holocaust research by providing online access to information about dispersed sources relating to the Holocaust (http://portal.ehri-project.eu). In this case, the problem at hand is to combine data coming from a network of archives in order to create an interoperable data space which can be used to search for, retrieve and disseminate content in the context of archival-based research. The scholarly purpose has specific consequences on our task. It assumes that the information made available to the researcher is as close as possible to the originating source in order to guarantee that the ensuing analysis can be deemed reliable. In the EHRI network of archives, as already observed in the case of the EU Cendari project, one cannot but face heterogeneity. The EHRI portal brings together descriptions from more than 1900 institutions. Each archive comes with a whole range of idiosyncrasies corresponding to the way it has been set up and evolved over time. Cataloging practices may also differ. Even the degree of digitization may range from the absence of a digital catalogue to the provision of a full-fledged online catalogue with all the necessary APIs for anyone to query and extract content. There is indeed a contrast here with the global endeavour at the international level to develop and promote standards for the description of archival content as a whole.

Nonetheless, in a project like EHRI, standards should play a central role. They are necessary for many tasks related to the integration and exploitation of the aggregated content, namely:

- Being able to compare the content of the various sources, thus being able to develop quality-checking processes;
- Defining of an integrated repository infrastructure where the content of the various archival sources can be reliably hosted;
- Querying and re-using content in a seamless way;
- Deploying tools that have been developed independently of the specificities of the information sources, for instance in order to visualise or mine the resulting pool of information.

The central aspect of the work described in this paper is the assessment of the role of the EAD (Encoded Archival Description) standard as the basis for achieving the tasks described above. We have worked out how we could develop a real strategy of defining specific customization of EAD that could be used at various stages of the process of integrating heterogeneous sources. While doing so, we have developed a methodology based on a specification and customization method inspired from the extensive experience of the Text Encoding Initiative (TEI) community. In the TEI framework, as we show in section 1, one has the possibility to model specific subsets or extensions of the TEI guidelines while maintaining

[1] https://team.inria.fr/almanach/

both the technical (XML schemas) and editorial (documentation) content within a single framework. This work has led us quite far in anticipating that the method we have developed may be of a wider interest within similar environments, but also, as we believe, for the future maintenance of the EAD standard.
Finally this work, successfully tested and implemented in the framework of EHRI [Riondet 2017], can be seen as part of the wider endeavour of European research infrastructures in the humanities such as CLARIN and DARIAH to provide support for researchers to integrate the use of standards in their scholarly practices. This is the reason why the general workflow studied here has been introduced as a use case in the umbrella infrastructure project PARTHENOS which aims, among other things, at disseminating information and resources about methodological and technical standards in the humanities.

1 Customizing and maintaining EAD with TEI-ODD

EAD maintenance issues

Developing international consensus on a standard for archival description is a daunting challenge. Cultural differences and established and differing theories and practices are at the core of the challenge. [EGAD ICA, 2016]

The challenge expressed by the ICA Experts group on archival description (EGAD) has been tackled since 1993 with the development of the Encoded Archival Description (EAD) [Library of Congress, 2013], which has successfully developed a standard format usable by a wide range of archives and archivists worldwide, making it possible to transcribe printed finding aids, as well as to describe archival records according to diverging national or institutional practices. However, since its creation, EAD faces criticism, as many observers are pointing to its permissiveness as a problem. Yet in 2001, Shaw asks for a "more prescriptive descriptive standard" [Shaw, 2001]. Still today, and even if EAD32 is globally seen as a step in the right direction, EAD is generally seen as a poorly structured and interoperable standard, not very suitable for data exchange, and is paradoxically considered by some information specialists, not a “standard for archival description” [Bunn, 2013]. In practical terms, each institution (each archivist), and each piece of software can have its own way of creating EAD, and the same material can be described in totally different ways. The first example that comes to mind is the choice to let the archivist use <c> or <c01>, <c02>, ... to describe sub-components. Therefore, it is important to document specific guidelines by institutions or from specific contexts (a thematic portal for example).
The agencies responsible for the maintenance of archival standards have developed several important initiatives in order to gain interoperability. EAD3, developed by an international subcommittee sipported by the Society of American Archivists[2], with the cooperation of many archivists worldwide, and maitained by the Library of Congress[3], is a big step towards interoperability even though many archives consider the change towards EAD3 as a mid-term perspective. On the other hand, since 2012, the International Council on Archives has been building the content model *Records in Context* [EGAD ICA, 2016], a descriptive standard that reconciles, integrates, and builds on its four existing standards: General International Standard Archival Description (ISAD(G)), International Standard Archival

[2] http://www2.archivists.org/groups/technical-subcommittee-on-encoded-archival-description-ead/encoded-archival-description-ead
[3] http://www.loc.gov/ead/

Authority Records - Corporate Bodies, Persons, and Families (ISAAR(CPF)), International Standard for Describing Functions (ISDF) and International Standard for Describing Institutions with Archival Holdings (ISDIAH). This initiative will also contribute to providing a solid framework for exchanging archival data more easily.
The maintenance of EAD, undertaken by the Society of American Archivists and the Library of Congress, is still a big issue. The maintenance of a standard requires in all cases discussions to achieve consensus, and any major revision should undergo a precise and complete process, although some minor corrections and adjustments are sometimes welcome in the meantime. Between EAD 2002 and EAD3, more than ten years passed and some features introduced in 2015 had been requested by the community many years before. For example, some users asked for a typing attribute for the `<ead:addressline>` element, a child of `<ead:address>` [EAD working group AFNOR]. This small modification was introduced in EAD3, as part of the general revision process, that lasted five years. One might have hoped for a smoother evolution of the standard, based on continuous maintenance, which is the case for the TEI consortium that updates its standard regularly on GitHub. In this respect, the fact that the development of EAD3 actually took place on GitHub opens the way to this more continuous maintenance.

Archive portals and EAD: use cases

The experience gained from concrete use cases has shown how strong the need is to build interoperability solutions between heterogeneous archival descriptions in EAD.
The Archives Portal Europe project (https://www.archivesportaleurope.net/), which brings together archival descriptions from all the European countries, has made an initial effort to implement common European profiles of EAD, EAC-CPF (Encoded Archival Context – Corporate Bodies, Persons and Families), EAG (Encoded Archival Guide) and METS (Metadata Encoding and Transmission Standard). Specific schemas were created, in particular apeEAD, a subset of EAD2002, which "was drafted on the basis of a comparison of EAD profiles and practices of the National Archives participating in the project"[4].
European funded research infrastructures and projects have also tackled this issue of interoperability of archival descriptions, with an additional focus on specific research communities, taking into account the specific needs of each. In the context of two H2020 Research infrastructure projects which manipulate archival data, Cendari (Collaborative European Digital Archival Research Infrastructure)[5] on medieval and modern European history and EHRI, various solutions were proposed. In the Cendari virtual research environment, where researchers have the possibility to select descriptions originating from various sources and create their own collections, EAD was customized with the addition of elements dedicated to the researchers' uses [Romary et al. 2017]. We should note here that the necessity to be able to customize the EAD model in order to add e.g. a more precise model for describing bibliographic sources [Medves, Romary, 2013] has been in tension in projects such as Cendari with a possible tendency to simply get rid of EAD as a reference format for integrating researchers' information and design an *ad hoc* format [Gartner 2015].
For the EHRI project, on which we will focus, the problem is slightly different. Researchers are not (yet) allowed to create, or pin and select, their own descriptions, but the heterogeneity of the archival descriptions, which have to be collected in a single pool and

[4] http://apex-project.eu/index.php/en/outcomes/standards/apeead
[5] http://cendari.eu

processed uniformly, make it necessary to create a straightforward workflow for the ingestion of archival data in the portal database. EHRI coordinates the activity of 24 institutions (research bodies, archives, libraries, etc.), but its archival portal hosts descriptions from around 1 900 institutions. An extra challenge for EHRI is that Holocaust archives are hidden, often dispersed over several institutions or several fonds. Moreover, EHRI intends to focus on Eastern Europe, where few archives are digitally advanced, due to lack of funding or technical infrastructures. These descriptions can be entered manually by EHRI staff, but the preferred method is semi-automatic ingestion of XML files in the database. The portal database is a graph database whose data model is based on the ICA standards – ISAD(G) and ISAAR-CPF, combined with extra administration fields.

Due to the variety of institutions providing data, EHRI has to deal with a great heterogeneity of data formats and of EAD flavours, even if EHRI has used EAD2002 since the beginning of the project in 2011. In 2015, a discussion arose about the opportunity to move towards EAD3, which had just been released, but the lack of visibility on its uptake in the archival community during the four years of the project made this choice too risky. The use of EAD in EHRI takes place at the two sides of the workflow: the ingestion and the export of archival descriptions. EAD2002 is the pivot format for the semi-automatic ingestion of data in the EHRI database. It is also used as an exchange format, with users being able to download any content of the portal in XML-EAD (or EAC and EAG for authorities and institutions). Therefore, there is a strong need for both valid and customized EAD (and EAC-CPF) schemes, for two kinds of tasks: The first one is the possible mapping of the data to XML-EAD if the descriptions are not provided in this format. The second one is a validity check to be sure that the EAD conforms to EHRI requirements.

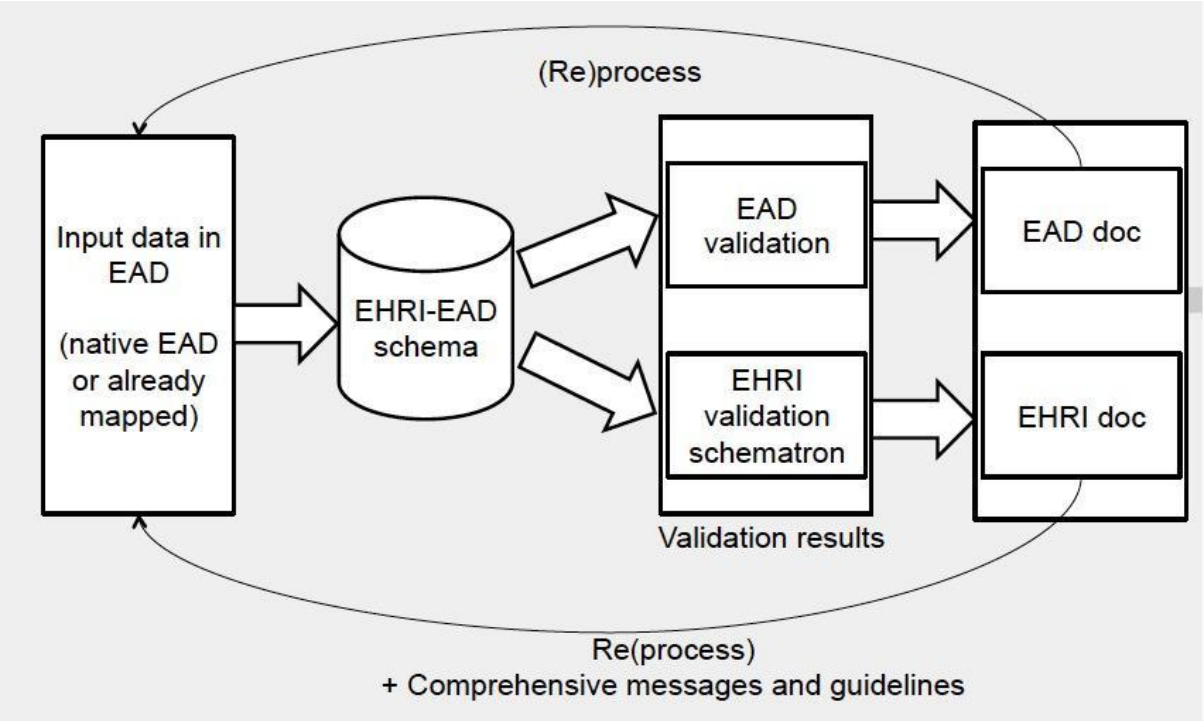

Figure 1: EHRI data validation workflow

The figure 1 above presents the workflow of archival data import in the EHRI portal. The archival materials provided by the institutions can be processed in two different ways. If they are not described in EAD2002, they are directly mapped to the customized EHRI EAD

format. If they are in EAD2002, we automatically check if the EAD flavour in input is compatible with the customized EHRI EAD format. If some adjustments are necessary, they can either be made by the provider himself of by the same EHRI mapping mechanism mentioned before. Then, the formatted XML document can be processed again in order to populate the EHRI database.

Project-oriented EAD schemas with TEI-ODD

There are several methods to create project-specific schemas. The most immediately obvious method is to modify the DTD or the schema by hand to narrow down the possibilities in a given context. We rejected this solution, however, as it is too restrictive, makes it harder to keep the history of changes, and, last but not least, schema validation errors provide a technical message, but not an archivist-oriented message. Another frequent problem is the absence of links between the online documentation (tag libraries, specifications) and the schema itself.

The solution we propose is based on a flexible and customizable methodology: it combines the complete description of the specifications in a machine-readable way, and customization facilities that are easy for the end-user (potentially non-specialists) to understand. More importantly, this solution doesn't change the core EAD schema, but adds more specific rules in a comprehensive and human-readable format, by combining the EAD schema (expressed in Relax NG) with ISO Schematron rules. Schematron is an ISO/IEC Standard (ISO/IEC 19757-3:2016) that parses XML documents and makes "assertions about the presence or absence of patterns"[6]. It can be used in conjunction with a large number of grammar languages such as DTD, relax NG, …

To do so, we made a natural choice, from a researcher's point of view, integrating EAD into an environment which is familiar to us, the Text Encoding Initiative (TEI), broadly recognized as the *de facto* standard for the representation of text in digital form. This choice is also coherent from an historical perspective, considering that the development of EAD was greatly influenced by the TEI. In more practical terms, one great quality of TEI lies in its capability to represent almost any digital resource. For instance, the TEI XML schema and the associated guidelines are maintained with the TEI format, more specifically, with a subset called "One Document Does it all" (ODD) which, as the name indicates, is a description language that "includes the schema fragments, prose documentation, and reference documentation [...] in a single document"[7], based on the principles of literate programming. Literate programming is a programming and documentation methodology whose "central tenet is that documentation is more important than source code and should be the focus of a programmer's activity" [Walsh 2002]. With ODD, semantic and structural consistency is ensured as we encode and document best practices in both machine and human-readable formats.

Originally, ODD relied on RelaxNG snippets encapsulated in TEI specifications, but has recenlty evolved towards a uniform language called PureODD, with a better power of expression and conciseness. However, the transition between these two methods is smooth and allow users to use both in one single specification, making it possible to move to PureODD gradually. ODD was initially created to give TEI users a straightforward way to customize the TEI schema according to their own practices and document this

[6] http://www.schematron.com/, accessed on November 2d, 2016.
[7] http://www.tei-c.org/Guidelines/Customization/odds.xml

customization. But it is possible to describe a schema and the associated documentation of any XML format, for example the W3C standard International Tag Set (ITS)[8] [Lieske et al. 2006]. Moreover, although ODD is a description language, it can be processed to generate an actual schema (a DTD, a Relax NG XML or compact schema and an XML schema), and documentation in various formats (XHTML, PDF, EPUB, docx, odt). We used ODD to completely encode the EAD standard, as well as the guidelines provided by the Library of Congress[9].

2 The EAD specification in ODD[10]

For clarification purposes, the code samples presented below come with prefixes : "rng" for Relax NG elements, "tei" for TEI(ODD), "ead" for EAD and "sch" for Schematron. The EAD ODD is a XML-TEI document made up of three main parts. The first one is, like any other TEI document, the `<tei:teiHeader>`, that comprises the metadata of the specification document. Here we state, among other pieces of information, the sources used to create the specification document in a `<tei:sourceDesc>` element. Our two sources are the EAD Tag Library[11] and the Relax NG XML schema[12], both published on the Library of Congress website. The second part of the document is a presentation of our method (the foreword) with an introduction to the EAD standard and a description of the structure of the document. This part contains some text extracted from the introduction of the EAD Tag Library.
The third part is the schema specification itself : the list of EAD elements and attributes and the way they relate to each other. As explained above, this specification can be expressed with RelaxNG snippets or in PureODD. In our case, we started the project while PureODD wasn't available, so most of the specification respect the former method, but we update it progressively, as a heterogeneous specification is perfectly functional. To understand the way ODD works, the most important elements are the following:

- Schema specification : `<tei:schemaSpec>`

The top-level ODD element is `<tei:schemaSpec>`. Its attributes are @start, which states "which patterns may be used as the root of documents conforming to it"[13] and @ns, for the namespace of the document.

- Element specification: `<tei:elementSpec>`

Each EAD element is described in a `<tei:elementSpec>` element, where the encoded information combines the element documentation in textual and machine interpretable form. These element declarations are connected to class declarations. In TEI, elements are members of one or more classes.

---

[8] https://www.w3.org/TR/its20/its20.odd, accessed on March 15th 2018.
[9] http://loc.gov/ead
[10] The EAD guidelines and schema encoded with ODD can be found here: https://github.com/ParthenosWP4/standardsLibrary/blob/master/archivalDescription/EAD/odd/EADSpec.xml, accessed on March 28th, 2017
[11] http://www.loc.gov/ead/tglib/index.html
[12] http://www.loc.gov/ead/ead.rng
[13] TEI Guidelines Version 3.1.0, element schemaSpec (schema specification), http://www.tei-c.org/release/doc/tei-p5-doc/en/html/ref-schemaSpec.html (accessed on January 12th, 2017)

<tei:elementSpec> combines technical specification elements (see below) and the documentation of the element. The most important documentation elements are <tei:gloss> and <tei:desc>. The former refers to a phrase or word used to provide a gloss or definition for some other word or phrase. <tei:gloss> contains the complete name of the element, as stated in the tag library. The latter is a brief description of the object documented by its parent element, typically a documentation element or an entity. In the EAD ODD, the value of <desc> is the first half of the tag Library description, which gives a formal definition of the element and which kind of information it can contain.
Other documentation like <tei:exemplum> and <tei:remarks> complete the element specification.

- Class specification: <tei:classSpec>

A class is "a group of elements which appear together in content models, or which share some common attribute, or both"[14]. Classes are defined by the <tei:classSpec> element. In our ODD specification, we encoded as classes the patterns defined in the EAD Relax NG schema. But Relax NG patterns and TEI classes behave differently. In Relax NG, the pattern gives the possible descendant nodes (elements, attributes, modules), whereas the <tei:classSpec> element declares its membership to upper modules. In other words, Relax NG has a top-down behaviour (patterns lists their members) and TEI ODD has a bottom-up behaviour (each element or class lists its membership to a class).
For instance, as presented in table 1, the pattern called "m.phrase.basic.norefs", which contains the pattern "m.phrase.bare" and the elements <ead:abbr> and <ead:expan>, becomes the class "model.phrase.basic.norefs", which have different content.

Relax NG	ODD
<rng:define name="m.phrase.basic.norefs"> <rng:choice> <rng:ref name="m.phrase.bare"/> <rng:ref name="abbr"/> <rng:ref name="expan"/> </rng:choice> </rng:define>	<tei:classSpec ident="model.phrase.basic.norefs" type="model" module="EAD"> <tei:classes> <tei:memberOf key="model.para.content.norefs"/> <tei:memberOf key="model.phrase.plus"/> <tei:memberOf key="model.phrase.basic"/> </tei:classes> </tei:classSpec>

Table 1: Specification of modules in Relax NG and ODD

The information contained in <rng:define name="m.phrase.basic.norefs"> is encoded in the element specification of <ead:abbr> and <ead:expan> and in the class specification of the class corresponding to the Relax NG module "m.phrase.bare" (table 2).

[14] TEI Guidelines Version 3.1.0, element classSpec (class specification), http://www.tei-c.org/release/doc/tei-p5-doc/en/html/ref-classSpec.html (accessed on January 12th, 2017)

```
<tei:elementSpec ident="abbr" module="EAD">
...
  <tei:classes>
    <tei:memberOf key="att.EADGlobal"/>
    <tei:memberOf key="model.phrase.basic.norefs"/>
  </tei:classes>
...
</tei:elementSpec>
```

```
<tei:elementSpec ident="expan" module="EAD">
...
  <tei:classes>
    <tei:memberOf key="att.EADGlobal"/>
    <tei:memberOf key="model.phrase.basic.norefs"/>
  </tei:classes>
...
</tei:elementSpec>
```

```
<tei:classSpec ident="model.phrase.bare" type="model"
module="EAD">
  <tei:classes>
    <tei:memberOf key="model.phrase.basic.norefs"/>
  </tei:classes>
</tei:classSpec>
```

Table 2: Module membership declaration in ODD

- Content declaration: `<tei:content>`

`<tei:content>` contains the machine-readable schema declaration of the content of the described element. It may be defined with a set of TEI ODD elements, or by using Relax NG patterns (the solution we use in this particular case). Some patterns contained in the Relax NG `<rng:element>` are copied and declared in the Relax NG namespace. This is the case for the patterns declaring which nodes are accepted as children of a given element.
Taking for example the element `<ead:unittitle>`, we can see (table 3) that most of the element declaration sequence is the same in the Relax NG schema and in the EAD ODD.

Relax NG	ODD
<element name="unittitle"> <ref name="a.common"/> <optional> <attribute name="label"/> </optional> <optional> <attribute name="encodinganalog"/> </optional> <optional> <attribute name="type"/> </optional> <zeroOrMore> <choice> <text/> <ref name="m.phrase.basic"/> <ref name="m.access"/>	<elementSpec ident="unittitle" module="EAD"> <!-- <tei:gloss> and tei:desc> --> <classes> <memberOf key="att.EADGlobal"/> <memberOf key="att.typed"/> <memberOf key="att.labeled"/> <memberOf key="model.data"/> <memberOf key="model.did"/> </classes> <content> <rng:zeroOrMore> <rng:choice> <rng:text/> <rng:ref name="model.phrase.basic"/> <rng:ref name="model.access"/> <rng:ref name="unitdate"/>

<ref name="unitdate"/> <ref name="num"/> <ref name="date"/> <ref name="bibseries"/> <ref name="edition"/> <ref name="imprint"/> </choice> </zeroOrMore> </element>	<rng:ref name="num"/> <rng:ref name="date"/> <rng:ref name="bibseries"/> <rng:ref name="edition"/> <rng:ref name="imprint"/> </rng:choice> </rng:zeroOrMore> </content> <!-- <tei:exemplum> and <tei:remarks> elements --> </elementSpec>

Table 3: Content declaration in Relax NG and ODD

- Definition of attributes

The available attributes for a given element are specified in a different way in ODD and in Relax NG. In ODD, the attribute definitions are always part of a list - `<tei:attList>` – that contains elements for each attribute – `<tei:attDef>`. However, ODD and Relax NG share the same use of data types declaration for attributes, ODD borrows some Relax NG elements, for instance `<rng:data>` (table 4).

Relax NG	ODD
<attribute name="otherlevel"> <data type="NMTOKEN"/> </attribute>	<attList> <attDef ident="level"/> <attDef ident="otherlevel"> <desc>...</desc> <datatype> <rng:data type="NMTOKEN"/> </datatype> <remarks>...</remarks> </attDef> </attList>

Table 4: Attribute definition in Relax NG and ODD

In the EAD Relax NG schema, attributes used by more than one element can be defined independently, and are then referenced with the element `<rng:attribute>` or `<rng:ref>` in the case where a group of attributes are defined together. In the EAD ODD, we created extra attribute classes for attributes that are used by more than one element, and use the class membership method to add attributes to an element (see table 5 for an example with `<ead:unittitle>`).

<rng:ref name="a.common"/> <rng:optional> <rng:attribute name="label"/> </rng:optional> <rng:optional> <rng:attribute name="encodinganalog"/> </rng:optional> <rng:optional> <rng:attribute name="type"/>	<tei:classes> <tei:memberOf key="att.EADGlobal"/> <!-- the class "att.EADGlobal" is similar to the Relax NG attribute module "a.common" + contains also the attribute @encodinganalog --> <tei:memberOf key="att.typed"/> <!-- class for the attribute @type --> <tei:memberOf key="att.labeled"/> <!-- class for the attribute @label -->

</rng:optional>	</tei:classes>

Table 5: Attributes declaration in Relax NG and ODD

3 Creating an EHRI-EAD schema

As previously stated, the power of ODD lies in its capacity for customization. It is possible and straightforward to create and document specific profiles (or EAD flavours) for an institution, a group of institutions (in a given country, for example) or a portal. For EHRI, we created another ODD to document the specific rules and constraints of the EHRI data model. In this new ODD file, the generic EAD specification is imported and serves as the baseline of specification. In the TEI environment, this is called chaining ODDs (Rahtz and Burnard 2014). The additional constraints are added only to the elements that they refer to. Therefore, the EHRI EAD ODD only contains the `<tei:elementSpec>` and `<tei:classSpec>` that are modified. The merge (or the chaining) of the two ODD files – the EAD generic and the EHRI specific – is made when we apply a transformation[15].

Typology of the constraints

The constraints that we need to add to EAD in order to ensure a smooth ingestion of descriptions in the database are of two types. First, some EAD elements are required for the correct functioning of the database, for instance unique identifiers for all the descriptions (i.e. the value of `<ead:eadid>`). Second, some elements are made mandatory for more qualitative reasons: for instance, to ease the discoverability of its resources, EHRI requires that a minimal description in English is provided with each description unit. Another example is the fact that EHRI encourages the use of ISO standards for the representation of languages, scripts, dates, etc, as well as the interlinkage of entities, via the use of authority lists.
Many specific validation rules were already used in EHRI, so we integrated them directly in the EHRI-EAD ODD[16]. The annexed table lists additional constraints found in the EHRI guidelines. Other constraints were found by EHRI database managers to ease the process of importing EAD documents into the database or were gathered by analysing samples from collection holding institutions (CHI). This approach will make it possible to ensure a very good quality of the EAD files, based on the very specific remarks made on relevant sample files.

All the constraints were sorted in categories, which we call roles. The different roles are:

- MUST: mandatory for the import process or according to the EAD (when we want to particularly highlight a requirement)
- SHOULD: mandatory for the description process, i.e. in terms of archival description. The SHOULD rules are not technically mandatory, but if they are not respected in the input description, it would be considered to be incomplete, with potential comprehension issues

[15] http://www.tei-c.org/Guidelines/Customization/odds.xml#body.1_div.2_div.7
[16] See them here: https://github.com/EHRI/data-validations/blob/master/schematron/ehri_ead.sch

- COULD: Non mandatory rules. This role combines the rules that would enhance the general quality of the description, without any obligation for the provider to follow the recommendation. They focus on the content-based element of `<ead:archdesc>`, pointing out that they could be added in the description, if they are not present in the input file.

This categorization is taken from the work previously done in EHRI around the preprocess of the EAD descriptions with the help of schematron rules[17].

Creating the customized EAD schema

Specific profiles are derived from the ODD master source described above (the generic EAD ODD). For each new EAD profile, a new ODD must be created. It must claim its inheritance to the master source, and modify the necessary specification elements, *i.e.* the `<tei:elementSpec>` and `<tei:classSpec>` that have a different behaviour. To change these behaviours, there are several solutions. The first and the simplest is to modify schema declaration elements: this means that the `<tei:content>`, the `<tei:attList>` or the `<tei:memberOf>` are directly modified.
Another solution, and the one we favour, is to use schematron rules, because they don't change the EAD schema and allow us to provide the user with comprehensive feedback. The rules are built with the element `<sch:assert>`, which means that the error message will be displayed when the pattern is not found.

Some rules reflect the requirements of EHRI database content model. For instance, it asks that the `<ead:date>` elements contain a `@normal` attribute whose content respects the ISO8601 standard on the representation of dates and time. This constraint is expressed in the ODD file with embedded schematron rules in the following way:

```
<elementSpec ident="date" module="EAD" mode="change">
    <constraintSpec ident="dateNormal" scheme="isoschematron" type="EHRI" mode="add">
       <desc>All the <gi>date</gi> elements MUST have a <att>normal</att> attribute whose
pattern respects the ISO8601 standard and take the following form: YYYY-MM-DD</desc>
           <constraint>
              <sch:rule context="ead:date">
                  <sch:assert role="MUST"
       test="matches(@normal,'^(([0-9]|[1-9][0-9]|[1-9][0-9]{2}|[1-9][0-9]{3}))-(0[1-
9]|1[012])-(0[1-9]|[12][0-9]|3[01])$')">@normal attribute MUST respect ISO8601 pattern =
YYYY-MM-DD</sch:assert>
              </sch:rule>
           </constraint>
       </constraintSpec>
</elementSpec>
```

The second rule is also a requirement, but for different reasons. In order to make the archival description understandable, EHRI requires that a `<ead:scopecontent>` element should be present somewhere. The choice is either for the provider to write general paragraph and put it at the highest level (`<ead:archdesc>`) or to add a more precise `<ead:scopecontent>` for each subcomponent, from `<ead:c01>` to `<ead:c06>`. Here,

[17] https://cdn.rawgit.com/EHRI/data-validations/master/schematron/rules.html

the rule is called at the <ead:archdesc> level, because CHI is more likely to provide a global <ead:scopecontent> if it didn't exist before.

```
<elementSpec ident="archdesc" mode="change">
<!-- … -->
    <constraintSpec ident="scopecontentInArchdescOrC" scheme="isoschematron" type="EHRI">
       <desc>A <gi>scopecontent</gi> element SHOULD be present in the description at least
       in <gi>archdesc</gi>, if not in the <gi>c</gi> elements.</desc>
       <constraint>
           <sch:rule context="ead:archdesc" role="SHOULD">
              <sch:assert test="ead:scopecontent or ead:dsc/ead:c01/descendant-or-
              self::ead:scopecontent">a "scopecontent" element SHOULD be present at least
              in "archdesc" if not in the "c" elements</sch:assert>
            </sch:rule>
       </constraint>
    </constraintSpec>
</elementSpec>
```

The last rule shown is the lowest level of constraint. It presents some possibilities to make the description more complete. In particular, these rules focus on the content related elements of <ead:archdesc>. Therefore, these messages are not considered as real errors, but as pieces of advice that the providers can choose to follow or not.

```
<elementSpec ident="archdesc" mode="change">
<!-- ... -->
    <constraintSpec ident="bibliographyPossible" scheme="isoschematron" type="EHRI">
       <desc>The <gi>archdesc</gi> element COULD contain a <gi>bibliography</gi>
                element.</desc>
           <constraint>
              <sch:rule context="ead:archdesc">
                     <sch:assert role="COULD" test="ead:bibliography">archdesc COULD have
                     a bibliography</sch:assert>
              </sch:rule>
           </constraint>
    </constraintSpec>
</elementSpec>
```

Use of the schema in EHRI's mapping and validation workflow

The schema created from the ODD file is used for the mapping and the validation processes of archival descriptions in the EHRI database. As the EHRI EAD schema is a Relax NG schema with embedded schematron rules, these combined languages are used for different parts of the process, in a two-step validation. The first step transforms the input files in EAD2002 if needed. In this case, the EAD schema used is the EHRI EAD schema, without the schematron rules. The second step is the validation with respect to the schematron rules. The schematron rules embedded in the EHRI-EAD schema are meant to be presented as a diagnosis to the content providers. This diagnosis will point to elements of the EAD that, even if they are in valid EAD, are not in line with the EHRI requirements. As we stated above, they are of three types :

- Some messages emphasize EAD validation errors by giving extra information,
- Some messages ask for modification in order to make the description compliant with the specific EHRI constraints,

- Some messages highlight some description elements that could be improved, but without any obligation to do so.

These three recommendation levels will correspond to three blocks of validation results, in order to show the users the elements they need to update in priority.
The validation against schematron returns a log containing the location (line and character in the line) of the error and a description (the message created in the ODD specification, see above). Following with the examples already explained, we show below a sample for each error severity (figures 2 - 4).

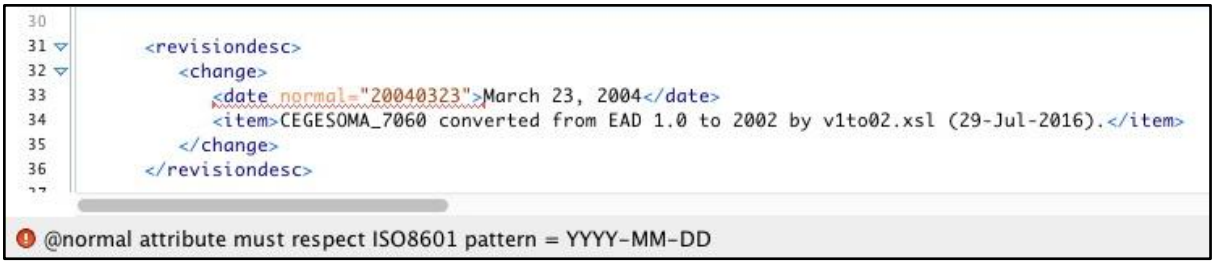

Figure 2: MUST constraint: Date normalisation rule

Figure 3: SHOULD constraint: Scope and content absence rule

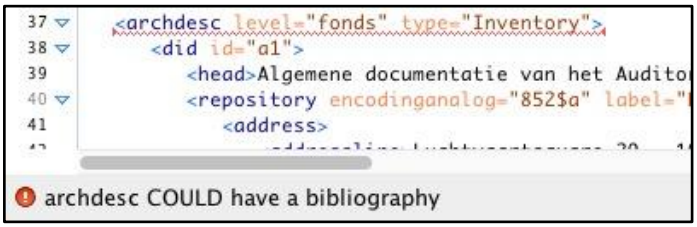

Figure 4: COULD constraint: Bibliography suggestion rule

A full description of the expected content (i.e. HTML "tag library") is generated from the ODD file. Ideally, the error message is displayed to the user with a link to the relevant section of the documentation. This could be implemented in Schematron with an extra attribute (that would likely be `@see`), and a stable URL template in which to interpolate this ID. Another implementation possibility would be, in some cases, to modify the input file on the fly, based on the results of the schematron validation. This solution is made possible by the Schematron Quickfix framework, that allows us to define fixes for schematron errors[18].

[18] http://www.schematron-quickfix.com/ accessed on March 28th 2017

Conclusion

This article lays the foundation of a solid framework for the customization and the maintenance of EAD, two of its acknowledged weaknesses. By using the possibilities offered by a well-established standard, the Text Encoding Initiative, to join the technical specification and the prose documentation in a single environment, we would like to pave the way for:

- A better understanding of what EAD can offer to its users community at large
- A straightforward way to maintain and improve this standard, integrating it with recent standardization evolutions and initiatives such as EAD3 and Records in Context
- A more consistent and extended use of EAD outside the archival community

We understand that these ambitions cannot be the responsibility of individual scholars and should be endorsed by a wider community and supported by research infrastructures, both of which can bring together enough knowledge and experience. We also advocate the adoption of such a framework by the other archival XML standards, in particular EAC-CPF (Romary, Riondet 2017), in order to obtain the result we are all striving toward: the largest use of the very precious content of archival data and metadata.

Acknowledgments

Special thanks to Annelies van Nispen (NIOD) and Hector Martinez Alonso (ALMAnaCH) for their help, and to Lou Burnard (TEI) for his wise comments.

References

- Archives Portal Europe network of excellence, D6.1 First Analysis report: Applying Web 2.0 solutions in archival applications, http://apex-project.eu/images/docs/D61_Web20_In_Archival_Applications.pdf, accessed on January 9th 2018
- Archives Portal Europe network of excellence D6.6 Second analysis report: Applying Web 2.0 solutions in archival applications, 2014, http://apex-project.eu/images/docs/D66_Web20_In_Archival_Applications_final.pdf, accessed on January 9th 2018
- Bunn, 2013. “Developing Descriptive Standards: A Renewed Call to Action.” Archives and Records 34 (2): 235–47. doi:10.1080/23257962.2013.830066.
- EAD and EAC-CPF working groups, AFNOR, Proposals for evolution of EAD, https://www2.archivists.org/sites/all/files/France_Proposals%20for%20evolution%20of%20EAD_0.rtf, accessed on January 9th 2018
- Experts group on archival description (ICA). “Records in Contexts, a Conceptual Model for Archival Description. Consultation Draft v0.1.” Conseil international des Archives, September 2016. http://www.ica.org/sites/default/files/RiC-CM-0.1.pdf, accessed on January 9th 2018
- Gartner, Richard. “An XML Schema for Enhancing the Semantic Interoperability of Archival Description.” Archival Science 15, no. 3 (September 1, 2015): 295–313. doi:10.1007/s10502-014-9225-1.
- Library of Congress, Development of the Encoded Archival Description DTD, 2013, http://www.loc.gov/ead/eaddev.html, accessed on January 9th 2018

- Lieske, Christian, Rahtz, Sebastian and Sasaki Felix, Internationalization and Localization of XML: Introducing "ITS", XTech 2006, Amsterdam, The Netherlands, May 2006, http://xtech06.usefulinc.com/schedule/paper/55, accessed on January 9th 2018
- Medves, Maud, Romary, Laurent. EAG(CENDARI): customising EAG for research purposes. *Building infrastructures for archives in a digital world*, Jun 2013, Dublin, Ireland. 2014. 〈hal-00959841v2〉
- Rahtz, Sebastian, and Lou Burnard. 2014. “Advanced Topics in ODD.” In ODD: One Document Does it All. Workshop at the Text Encoding Initiative Conference and Members Meeting, Oct 22–24 Evanston, IL. http://tei.it.ox.ac.uk/Talks/2014-10-odds/talk-05-advanced.xml
- Riondet, Charles, Romary, Laurent, Van Nispen, Annelies, Rodriguez, Kepa Joseba, Bryant, Mike. Report on Standards. [Contract] D.11.4, Inria Paris. 2017. 〈hal-01503235〉
- Romary, Laurent, Riondet, Charles. Ongoing maintenance and customization of archival standards using ODD (EAC-CPF revision proposal). EAC-CPF revision proposal. 2017. 〈hal-01677185〉
- Romary, Laurent, Banski, Piotr, Bowers, Jack, Degl’innocenti, Emiliano, Ďurčo, Matej, et al.. Report on Standardization (draft). [Technical Report] Deliverable 4.2 Inria. 2017. 〈hal-01560563〉
- Shaw, Elizabeth J. 2001. “Rethinking Balancing Flexibility and Interoperability.” New Review of Information Networking 7 (1): 117–31. doi:10.1080/13614570109516972
- Walsh, Norman , Literate Programming in XML, 2002, http://nwalsh.com/docs/articles/xml2002/lp/paper.htmlaccessed on January 9th 2018