COMEDI: A new component metadata editor

Gunn Inger Lyse*, Paul Meurer†, Koenraad De Smedt‡
University of Bergen*,‡, Uni Research Computing†
Bergen, Norway
E-mail: gunn.lyse@uib.no*, paul.meurer@uni.no†, desmedt@uib.no‡

Keywords: metadata, CMDI, editor

1. Introduction

Metadata for language resources and tools enhance their visibility, reusability and durability (Trippel et al., 2014; Piperidis et al., 2014; Dima et al., 2012; Lyse et al., 2012). The Component Metadata Initiative (CMDI) (Broeder et al., 2010) has led to a standard with the benefit of modularity through reusable components and standard profiles.

Still, the process of creating and managing metada is a challenge for the less experienced and time consuming even for the experienced. We therefore present the design and implemention of COMEDI (COmponent Metadata EDItor), a new CMDI-compatible editor, motivated by the wish to offer even more efficiency and user-friendliness. COMEDI can start from any metadata schema conforming to CMDI. Among its features are an intuitive web interface, cloning of information from existing metadata files and autofilling, such as the ISO code for languages.

The editor has been created in the context of CLARINO, which is implementing the Norwegian part of the CLARIN infrastructure. The metadata work in CLARINO builds on the work done in the META-NORD project (2011-2013), in which the Norwegian partners produced 67 metadata files describing a range of language resources made available in Norway (Haltrup Hansen and Pedersen, 2013). The metadata was published on META-SHARE (see Section 2.). The META-SHARE model has been mapped and incorporated into the Component Registry of CMDI (Piperidis et al., 2014). As will be discussed in Section 2., META-SHARE and other existing editors have advantages but also drawbacks, which have motivated the development of COMEDI.

2. Existing metadata editors

2.1. Arbil

Arbil[1] is a metadata editor, browser and organizer tool for CMDI, IMDI and similar metadata formats (Withers, 2012). It is characterized as 'the reference implementation for a CMDI editor', but is primarily directed towards archivists or librarians rather than non-expert researchers; the latter group may therefore find the learning curve in Arbil rather steep (Dima et al., 2012).

Arbil allows users to create and edit metadata displaying the underlying XML-code as plain table structures. Arbil offers several facilities to reduce the burden of repetitive typing tasks: it allows editing of metadata via copy and paste into multiple fields of multiple rows (bulk editing); moreover, frequently used sections of metadata can be collected from a 'favorites' directory and be reused for new metadata. The editor warns the user when a metadata field is missing or is not in the required format but allows the user to continue editing and to save even with errors.

The downside of Arbil, however, is its steep learning curve, which makes it less user-friendly for the targeted average metadata provider within CLARIN; moreover users report that they experience Arbil as responding very slowly.

2.2. ProFormA

ProFormA[2] is a web-based CMDI editor (Dima et al., 2012). Through a web interface, the user selects a CMDI profile to start from, and the editor displays the profile as a plain online form, hiding the XML code. The user may create new files, upload and edit existing files and download the result as an XML file.

ProFormA is mainly designed to support NaLiDa profiles and users, and does not seem to adequately support the full CMDI framework.[3] Among other things, there are display errors related to cardinality: with elements that may occur zero or infinitely many times only one instance can be created. Conversely, with elements where the CMDI profile allows zero or at most one item infinitely many instances are allowed.

Moreover, the hierarchical structure is not always correctly displayed. Sometimes the elements of a subcomponent are displayed without a subcomponent title, which leaves the subcomponent elements completely out of context. This is particularly unfortunate since ProFormA also omits the documentation text that ususally accompanies elements and components in the CMDI profile.

There are also limitations regarding controlled vocabulary. ProFormA allows the user to choose language names from a drop-down menu only listing four languages: Dutch, English, French, and German. It is possible to type other language names manually, but since ProFormA does not validate content provided by the user, any string (even ill-formed) will pass.

In conclusion, ProFormA currently appears to be of limited use for the full choice of CMDI profiles in the Component Registry.

[1] `http://tla.mpi.nl/tools/tla-tools/arbil/`

[2] `http://www.sfs.uni-tuebingen.de/nalida/proforma/web/`

[3] We tested by uploading the profile *resourceInfo (META-SHARE v3.0 – lexical/conceptual resources)*; ID: clarin.eu:cr1:p_1355150532312

2.3. META-SHARE

The META-SHARE model is tailored to describe language resources and tools relevant for language technology R&D, and thus has not been created to support the full CMDI framework. It offers metadata schemas for four resource types: corpus, lexical/conceptual resource, tool/services and language description. Metadata can be created, stored, edited and updated using an online META-SHARE editor which is implemented as a web-based form.[4] Metadata files can be converted to CMDI, e.g. using XSLT, since the four META-SHARE schemas have CMDI counterparts in the Component Registry (Piperidis et al., 2014).

META-SHARE stores files which are then visible and accessible to the resource owners and the defined editing group for easy cooperation between several individuals.

META-SHARE offers differing degrees of descriptive detail at two levels: the minimal schema contains obligatory elements (e.g. resource name and a resourceType classification), and the maximal schema contains optional information. META-SHARE does not allow the user to save a copy (even temporarily) unless the information for the minimal schema is complete and valid.

The editor offers autofill (e.g. for today's date) and features controlled vocabularies extensively through drop-down lists (e.g. linguality type) and autocompletion (e.g. language codes), which promotes consistent and correctly typed metadata. The amount of repetitive typing is greatly reduced by storing person info, institutional info and research project info as separate objects in a database which can be pointed to by multiple metadata files. If information about an object must be changed later (e.g. changing the e-mail of a person object), the change is automatically updated in all metadata files pointing to this object.

A challenge with large metadata schemata is how to portion out elements, components and subcomponents, and META-SHARE appears as very complex and at times confusing in this respect. Subcomponents are usually shrunk initially, but the editor misses a uniform display both for shrunk and expanded components and for how to expand components[5].

In conclusion, the META-SHARE editor has user-friendly features; however, like ProFormA, it has not been designed to fully support CMDI profiles, which is a non-trivial drawback. There is also a potential for other improvements.

2.4. Other editors

General purpose XML editors such as Oxygen[6] are challenging to use for non-experts, since their use requires some insight in the XML technology (Dima et al., 2012). The IMDI-editor[7] seems to be more or less replaced by Arbil and will not be further discussed in this paper. CLARIN-D has created the HTML5 web app CMDI Maker,[8] which is now part of the CLARIN infrastructure and which allows the user to create IMDI files that may subsequently be uploaded as CMDI with IMDI profile via Arbil.

3. COMEDI

3.1. Functionality and features

COMEDI offers a web interface.[9] It is currently integrated in a web framework at the emerging Clarin B-center in Bergen, but can also be deployed as a stand-alone service.

COMEDI handles any CMDI-compatible profile (version 1.1; version 1.2 under development). Similarly to ProFormA, the user can create new files by typing the CMDI profile ID (cf. Figure 1), or by choosing between a subset of available profiles in a drop-down menu. The user then chooses a file identifier (in Figure 1 the name 'demo-corpus-profile' was entered). This creates a new, empty file based on the selected CMDI profile.

Figure 1: Screenshot illustrating the creation of a new metadata file based on a CMDI profile (referring to its profile ID).

The user may also upload and edit existing metadata records. A metadata file in COMEDI can be exported as a CMDI record, or embedded in an OAI-PMH wrapper.

A metadata schema may appear overwhelming, and the ability to hide elements is therefore beneficial. COMEDI displays one top-level component at a time while keeping a component menu at the top of the page where the user may switch to another component (cf. the top line menu in Figure 2, where the currently chosen component, *Contact person* is displayed in boldface).

COMEDI offers advanced navigation functionality: all operations, e.g. navigating from one component or element to another, switching between edit and view mode, editing content, adding or removing components and elements, showing and hiding subcomponents, or switching between top-level components, can be done with keyboard shortcuts alone or by mouse-clicks. This should accommodate both occasional and regular users. An on-line wiki-type documentation is provided.

In COMEDI there are two display modes, *view* mode and *edit* mode. In view mode, the editor displays in red if obligatory elements are missing in the given component or if user-provided content does not validate correctly. For instance, Figure 2 shows the *Contact person* component in view mode, where the obligatory e-mail address is missing. In edit mode, the user may enter metadata in any order desired. In both modes, the components and their elements are shown hierarchically as boxes containing boxes (cf. Figure 2). Components can be expanded and shrunk by a mouse click or a keyboard shortcut, akin to the profile display in the Component Registry. Initially, all but the top-

[4] Downloadable from https://github.com/metashare/META-SHARE/downloads

[5] See for instance the discussion at: https://github.com/metashare/META-SHARE/issues/315

[6] http://www.oxygenxml.com/

[7] https://tla.mpi.nl/tools/tla-tools/older-tools/imdi-editor/

[8] http://class.uni-koeln.de/cmdi_maker/

[9] http://clarino.uib.no/repository

Figure 2: Displaying one top-level component at a time (view mode).

level components are shrunk. Optional uninstantiated components are grayed out but visible in edit mode and hidden in view mode.

Each component and each element is displayed along with its documentation from the Component Registry. Adjacent to the name, the number of instantiations of a component or element is given, along with the allowed minimum and maximum. When the minimum cardinality equals 1, the editor by default instantiates this component once, even though it is initially empty. For instance, the component *Person info* occurs once and must occur exactly once, hence we see: [1/1] in Figure 3. Instances of elements and components can be created or deleted using [+] and [−] buttons, if allowed by the component definition.

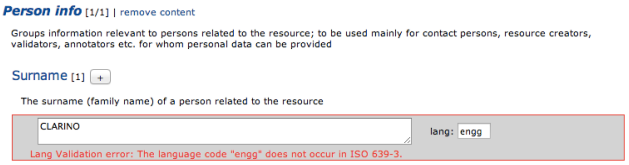

Figure 3: Edit mode: validation of language code on the fly.

The editor saves working copies during a working session in the server database, so that work can be resumed at any time. This functionality works regardless of whether all user content is valid according to the profile specification. As any modern web-based editor, COMEDI supports Unicode character input. Spell checking is handled by the web browser.

Validation, controlled vocabulary and autocompletion are indispensable tools to reduce the amount of inconsistencies and errors in the metadata, while also saving time for the metadata creator. The COMEDI editor validates input according to the ValueScheme specification in the element definition and displays an error message on invalid input. Support for vocabulary services like OpenSKOS is planned in the transition to CMDI version 1.2. autocompletion and autofill is available where appropriate (e.g. autofill in today's date in MetadataLastDateUpdated).

COMEDI allows easy cloning of components from existing metadata records stored in the repository, thus greatly reducing the work burden of repetitive typing tasks. When clicking on *select component* next to the component name, a list of existing components of the same type (ComponentId) appears (Figure 4). The user can scroll through the list. Upon selecting one of the items, its content is copied into the component in focus. To make this feature safe to use, a component can only be filled with new content if it is empty; otherwise, it has to be cleared first. Similarly to Arbil, useful components can be marked as favorites in the component selection box; they will then appear first next time the user evokes the component list for this component type. Cloned contents may be edited as desired. Such editing will not affect the contents of the original, and the edited version will in that case simply appear as a new item in the list of components.

A similar but distinct feature is the availability of instantiated components that can be pointed at from different places, in the sense of structure sharing. When such a component's content is changed in one place, the changes will be reflected in all metadata records referring to it. This feature, which is also provided in the META-SHARE editor (cf. Section 2.3.), is particularly useful for components describing person or institutional info and the like, where changes should be propagated *passim*.

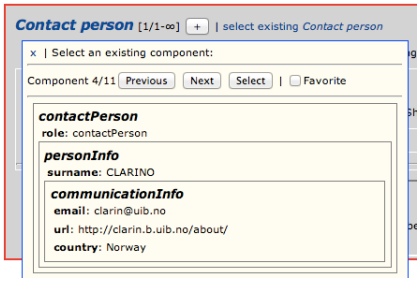

Figure 4: Cloning components: selecting an existing component for cloning.

COMEDI has support for the id/ref mechanism in CMDI. The Resources section of the metadata record can be edited in much the same way as ordinary components. Resource proxies can be referred to in components, and the consistency of the ids and references is checked. The implementation of other Resource elements is planned.

The contribution of metadata by unknown users is generally undesirable. Users of the editor (as of other resources hosted at the center) are therefore authenticated via the CLARIN IdP, the eduGAIN interconnection of IdP federations, or OpenIdP (provided by Feide). Authorization with differing degrees of rights can be given on an individual basis.

3.2. First user evaluation

Even if a clean web interface and features such as autofill and cloning are known to add to user-friendliness, the actual user experience is an empirical issue. As a pilot study, COMEDI was tested on a researcher of linguistics with high technical skills in general, but without previous experience with metadata. The researcher was asked to fill in metadata for a lexical resource that he knows well. Overall, the test person found that the threshold for using COMEDI was low: getting started was easy and it was easy to keep track of the editing process thanks to the top-level component menu at the top of the page and the effortless shift between view

and edit mode. The user identified some weaknesses which will easily be improved in the next version of COMEDI. For instance, the researcher missed an autosave function when navigating from edit to view mode. Also, it proved confusing that the action of clicking on a title causes different things to happen depending on whether it is an element title (sending the user to the ISOcat page documenting that element) or a component title (shrinking or enlarging the component by clicking on it). To avoid this confusion, the ISOcat link will instead be available as an explicit link next to the title. We are planning a wider and more systematic evaluation of the usability of COMEDI.

3.3. Implementation

COMEDI is written in Common Lisp, like the other advanced tools in the INESS and CLARINO infrastructures at the Bergen center. The metadata being edited are stored in a relational database. Central to all operations performed on metadata in COMEDI are the CMDI profiles. The latter can be fetched from the Component Registry in the XML-based CMDI Component Specification Language (CCSL). Schema descriptions of metadata are derived from the CCSL format. The main idea in the implementation of COMEDI is to keep the profile, as a description of possible metadata, tightly connected to the metadata as a valid instantiation of the profile. In concrete terms, both the profile and the (complete or emerging) metadata are aspects of the same in-memory tree representation.

To start with, the profile (in CCSL format) is parsed into a DOM tree using a DOM parser. The DOM parser is however modified in such a way that it adds specific wrapper nodes around the CMD_Component and CMD_Element nodes of the profile: first, each CMD_Component is wrapped into a CMD_Component_Wrapper. Its cardinality attribute is set to 0 or 1, in accordance with the value of the component's CardinalityMin. Then, each CMD_Element is wrapped into a CMD_Element_Wrapper. If the element's CardinalityMin is 1, a CMD_Element_Realization is appended as a child node of the wrapper after the CMD_Element node.

This is the state when the metadata is still empty, or, more precisely, minimal. All components that have to be instantiated have been given cardinality 1 in the wrapper, and all elements that have to be instantiated are represented as a CMD_Element_Realization, but their values are empty.

Editing of the metadata is reflected in changes in the DOM tree: When an instantiation of an element is added in the editor, a new CMD_Element_Realization node is created. When a component is added in the editor, the wrapper's cardinality is increased, and unless the new cardinality equals 1, the CMD_Component node itself is cloned and added as a new child node to the wrapper. Editing of values simply results in changing the element realization's value attribute. New values are immediately validated against the CMD_Element. Basically the same operations are executed when existing metadata is read.

Starting from the unified DOM tree representation of both the profile and the metadata, the HTML and Javascript/AJAX code of the editor can be generated in a quite straightforward way: The DOM tree is serialized to XML, and appropriate XSL and CSS stylesheets create the HTML code. Since all component and element information of the profile is available in the XML, the stylesheets can create the necessary buttons and input elements to manipulate the metadata, and specifically, will only create those buttons and elements that are in accordance with the profile and result in admissible manipulations.

4. Conclusion and future work

The development of COMEDI is well motivated since it offers several advantages over existing editors, above all a clear but highly functional web interface abstracting away from technical details in an elegant manner while still keeping the internal structure of the metadata explicit, thus helping to produce metadata faster and more consistently (component cloning, autofill, validation of user input). It also features advanced navigation through keyboard shortcuts. Equally important is its support for the full CMDI v. 1.1 specification.

The editor is functional but will benefit from further testing for continued development. Among other things, we foresee the need to improve the search possiblities in metadata. Also, user experiences need to be further evaluated. COMEDI will be available in the public domain under a BSD license.

5. Acknowledgements

The research reported in this paper has received support from the Research Council of Norway through CLARINO.

6. References

Broeder, D., Kemps-Snijders, M., Uytvanck, D. V., Windhouwer, M., Withers, P., Wittenburg, P., and Zinn, C. (2010). A data category registry- and component-based metadata framework. In *LREC'10*, pages 43–47, Valletta, Malta. ELRA.

Dima, E., Hoppermann, C., Hinrichs, E., Trippel, T., and Zinn, C. (2012). A metadata editor to support the description of linguistic resources. In *LREC'12*, Istanbul, Turkey. ELRA.

Haltrup Hansen, D. and Pedersen, B. (2013). Deliverable D3.3 third batch of resources including resources selected in D2.4. Technical report, META-NORD.

Lyse, G. I., Escartín, C. P., and De Smedt, K. (2012). Applying Current Metadata Initiatives: The META-NORD Experience. In *Describing LRs with Metadata: Towards Flexibility and Interoperability in the Documentation of LR (LREC'12 Workshop)*, pages 20–27. ELRA.

Piperidis, S., Papageorgiou, H., Spurk, C., Rehm, G., Choukri, K., Hamon, O., Calzolari, N., Gratta, R. D., Magnini, B., and Girardi, C. (2014). Meta-share: One year after. In *LREC'14*, pages 1532–1538, Reykjavik, Iceland. ELRA.

Trippel, T., Broeder, D., Durco, M., and Ohren, O. (2014). Towards automatic quality assessment of component metadata. In *LREC'14*, pages 3851–3856, Reykjavik, Iceland. ELRA.

Withers, P. (2012). Metadata Management with Arbil. In *Describing LRs with Metadata: Towards Flexibility and Interoperability in the Documentation of LR (LREC'12 Workshop)*, pages 72–76. ELRA.