Originally published in: Calzolari, Nicoletta/Choukri, Khalid/Declerck, Thierry/Doğan, Mehmet Uğur/Maegaard, Bente/Mariani, Joseph/Moreno, Asuncion/Odijk, Jan/Piperidis, Stelios (Eds.): Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12). May 21-27, 2012, Istanbul, Turkey. – Paris: European Language Resources Association, 2012. Pp. 1061-1066.

A Metadata Editor to Support the Description of Linguistic Resources

Emanuel Dima, Erhard Hinrichs, Christina Hoppermann, Thorsten Trippel and Claus Zinn

Department of Linguistics, University of Tübingen
Wilhelmstr. 19, 72074 Tübingen, Germany
firstname.lastname@uni-tuebingen.de

Abstract

Creating and maintaining metadata for various kinds of resources requires appropriate tools to assist the user. The paper presents the metadata editor *ProFormA* for the creation and editing of CMDI (Component Metadata Infrastructure) metadata in web forms. This editor supports a number of CMDI profiles currently being provided for different types of resources. Since the editor is based on XForms and server-side processing, users can create and modify CMDI files in their standard browser without the need for further processing. Large parts of ProFormA are implemented as web services in order to reuse them in other contexts and programs.

Keywords: Component Metadata Infrastructure (CMDI), metadata editor, XForms

1 Introduction

Describing primary research data by metadata is a challenging task when resources are archived, especially given the existing variety of metadata schemas. For non-experts, the learning curve for schemas and the technology to create and maintain metadata is steep. In this paper, the metadata editor *ProFormA* is introduced that supports non-expert metadata providers in creating highly structured data with an easy to use interface implemented as a web form. The editor adjusts the web form to the potentially very different metadata schemas used for describing language resources within the Component Metadata Infrastructure (CMDI) framework (Broeder et al., 2010).

The Tübingen initiative for the sustainability of linguistic data (NaLiDa, see www.sfs.uni-tuebingen.de/nalida) has defined various CMDI profiles each accounting for one class of resource (lexical resource, corpus, tool, experimental data, etc., see also (Barkey et al., 2011a)). CMDI profiles provide XML Schemas (Thompson et al., 2004), expressing a set of rules to which an XML-encoded metadata document has to conform to in order to be considered *valid* according to that schema. The task of creating a metadata document should be completed by the researcher – as the person who created the resource in the first place and the one who is best qualified to describe it. To be able to provide such an XML file, the researcher needs predefined profiles that can be selected for a given resource at hand and tools to support the creation process. Although there are tools available for the provision of metadata, these do not meet the requirements of the CMDI creation process within the NaLiDa project. As the project's target user group includes researchers (mainly linguists) who are neither experienced with XML technologies nor with aspects concerning archiving or metadata creation, the need of finding another solution arose. On the basis of these insights, the decision was taken to develop a project-independent metadata editor, which supports the description of linguistic resources.

This paper gives an account on the metadata editor that is currently being developed. The editor aims at hiding most of the complexities of the CMDI framework and its embedding into XML technologies. It strives for keeping the provision of metadata as simple as possible while ensuring that all information required is entered and valid. The editor is designed, thus, to best support NaLiDa profiles and users, rather than to support the full CMDI framework and its large and heterogeneous user groups.

The remainder of the paper is structured as follows: related work is introduced in Sect. 2. In Sect. 3, the editor's design requirements are listed, serving as the foundation of the implementation. Sect. 4 describes the editor's technological underpinnings and its current status. Finally, Sect. 5 concludes and gives an outlook on future work.

2 Background / Related Work

For describing language resources, there are quite a few metadata conventions to choose from, ranging from the general Dublin Core metadata set (Coyle and Baker, 2009) to systems specifically designed for the adequate description of language resources such as OLAC (Simons and Bird, 2008) and IMDI (IMDI, 2001). Recently, CMDI has gained considerable traction and community support (Broeder et al., 2010). In the CMDI framework, elementary descriptors (i.e., *metadata fields* or *data categories*) are defined, centrally stored, and managed in the ISOcat registry (ISO 12620:2009). These data categories can be grouped together into *components*, which can then be shared using the Component Registry (see http://catalog.clarin.eu/ds/ComponentRegistry/#). Components are the building blocks of *metadata profiles*, which are templates for the description of one type of linguistic resource; each of its descriptional constituents can be traced to either ISOcat or the Component Registry.

In order to create resource descriptions, general purpose XML editors such as Oxygen (http://www.oxygenxml.com/) are hard to use, as they require intricate knowledge about XML technologies. Arbil (http://www.lat-mpi.eu/tools/arbil), on the other hand, is a metadata editor which is directed more towards archivists or librarians rather than individual researchers who only occasionally provide metadata. Those researchers might experience a rather steep learning curve in Arbil. Nevertheless, Arbil is the reference implementation for a CMDI editor and supports the complete CMDI

functionality.

3 Requirements

The design of the metadata editor aims at assisting individual researchers in efficiently describing linguistic resources with metadata according to the CMDI framework. The most important user requirements include that the editor should:

- support users in selecting the most adequate profile from a given set of predefined profiles, given the type of linguistic resource that needs to be described (the so-called 'profile wizard').
- give a form-based view of the chosen metadata profile; the form should resemble paper-type forms that segment the profile into well-defined and logically-related chunks. Ideally, the names of the fields will enable the user to understand the purpose of the field providing essential information on the expected content.
- offer context-sensitive help, based on the definitions of data categories in the ISOcat registry. Thereby, users will be supported during the process of filling-in the values for form elements. This feature will also prevent users from tag abuse in cases where the meaning of form elements is not clearly evident from their names.
- resemble a summarized technical data sheet for each resource. Such a data sheet is used by the overview for each resource in the NaLiDa faceted browser (Barkey et al., 2011a; Barkey et al., 2011b), i.e. an HTML-rendered overview of a resource's metadata, which is originally represented in XML.
- make provisions for auto-completion or pick-list selections for each field descriptor whose value range is defined with the help of a controlled vocabulary. Here, free-text input will be discouraged.
- validate all content provided by the user given the respective definitions of the data categories filled in. This includes the use of closed vocabularies, numeric formats, strings, and other data types.

In the implementation process, these requirements will be targeted in the outlined order, reflecting the prioritization used. Additional features that would be useful to have are, for example, a progress bar or a similar feature, indicating the minimal level of adequate description to be achieved for a given resource type. For this purpose, the editor needs to distinguish obligatory from optional fields and enforce that obligatory entries are filled out. Incomplete forms can be saved but will not be validated. Moreover, the editor should highlight important fields and entice users to supply such information. This is intended for a later state of development.

4 Technological Underpinnings

ProFormA is a suite of tools, interfaces and services providing the metadata editor, which is illustrated in Figure 1. Its components can also be used independently of the editor. The editor's technology is based on existing XML standards and tools (in particular, XForms, see (Boyer, 2009)) and is embedded into the CMDI framework. The CMDI Component Registry is used to distribute the component specification and the XSchema for the profiles. The tools for processing XForms consist of libraries transforming XForms into XHTML and JavaScript for browser handling. To fulfill the aforementioned design requirements, information sources are needed in addition to CMDI profiles. These are called *(profile-specific) editor configuration files* (see Section 4.3).
Most components are implemented as web services, enabling easy integration into other applications, such as repositories and archiving tools (for example, a tool as described by (Dima et al., 2012)). For standalone editing and viewing, a web application was designed to provide access to the CMDI files and to edit the forms.

4.1 Use of the Existing (External) Infrastructure

The metadata editor relies on the availability of the CMDI profiles, which are provided by the Component Registry (see `www.clarin.eu/cmdi`). Access to the CMDI profile is based on the REST interface of the registry. Consequently, the profile can be retrieved by a unique URL. The Component Registry used by ProFormA delivers two different kinds of representations: the XSchema, included in validating instances, and a description of the profile in a specific XML format, the CMDI Component Specification Language (CCSL). Internally, the Component Registry relies on the CCSL to generate the XSchema. Hence, the CCSL is the reference description.

4.2 CMDI XML Skeleton Generator

The CMDI skeleton generator is a web service generating an empty CMDI instance based on the CCSL definition of a CMDI profile. It provides an XML structure which shows the required and available elements. General XML editors and others hiding the XML complexity make use of the generated instance.

4.3 Profile-Specific Editor Configuration

Some features of the structured layout of an editor are independent of the underlying data structure. The goal of simplicity for the user conflicts with this independence because it could result in double maintenance. To avoid double maintenance, an intermediate layer is introduced. This layer contains either a default layout structure generated from the data structure, or an altered layout structure modified according to the requirements of the interface. If required, the modification is performed based on the user evaluating the resulting form.
For generating the editor, ProFormA uses the CCSL profile descriptions, which also allow to process unknown profiles. The profile description and the previously created CMDI metadata instance are displayed together in the editor, taking the values of the fields that are already available.

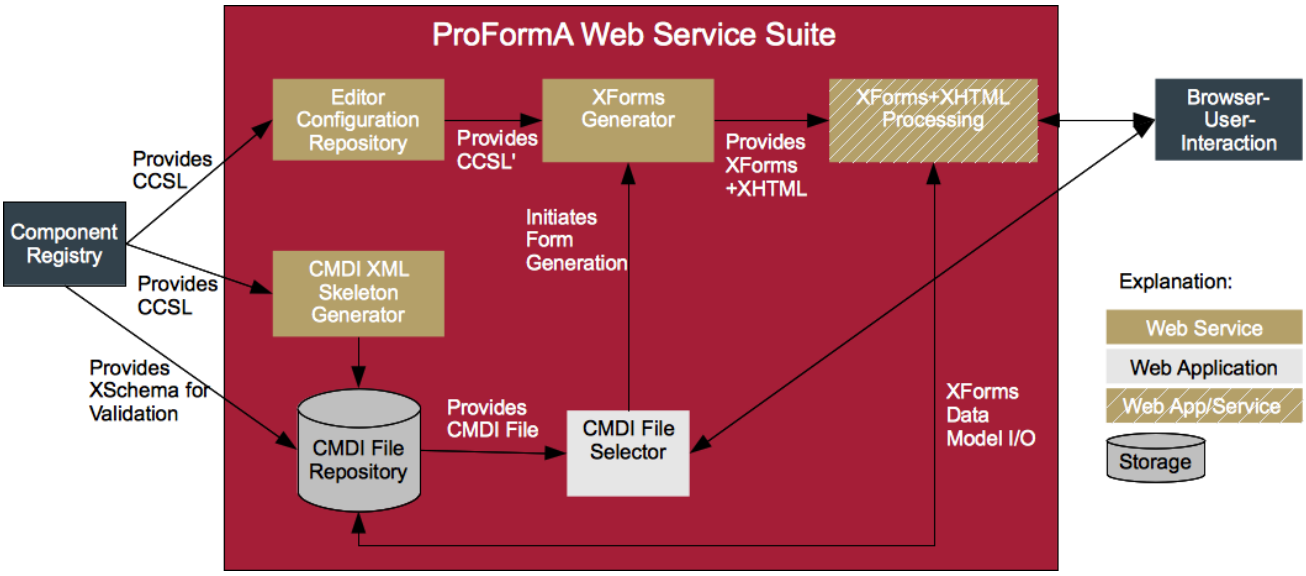

Figure 1: ProFormA CMDI editor suite of web services.

These values can include static information on the provider such as the provider's name and affiliation. The instances are either created by the skeleton generator or by another CMDI instance producing process, such as the transformation from other metadata formats.

Changes to the default profile descriptions (i.e. the CCSL) are made to a copy of the CCSL. These changes, if wanted, are applied to the configuration file manually, creating an extension to the CCSL, here written as CCSL'. The editor configuration repository is a system with two functions (cf. Figure 1): If the further processes request an unknown profile, it serves as a proxy to retrieve the profile's CCSL file from the Component Registry. But if the processes try to gain access to known profiles, the local copy is distributed. These local copies may contain the following modifications:

- *Grouping* of metadata fields in the editor's display deviating from the default grouping which is based on the underlying CMDI component structure. Thus, the form specification may produce a layout structure that is different from the order of elements in the CMDI profile. For example, it can be advisable to group a person and its institution together, though this might not be reflected in the CMDI components.

- An *editing priority* for each metadata field indicating if this field is `required`, `recommended` or `optional`.

- A *display priority* for each metadata field specifying if it should be displayed with `high`, `medium` or `low` priority. This option can be applied, for instance, to display different amounts of data or to hide empty fields that have a low priority.

- *Complimentary help text* for field elements. Some definitions for the intended use of the fields are already available by integrating the definitions both of the data categories in ISOcat and the components in the Component Registry. However, if additional hints should be provided, this can also be included in the configuration files.

4.4 XForms Generator

The metadata editor is entirely based on XForms (Boyer, 2009), a W3C standard for the definition and management of web-based forms. The XForms generator is a web service creating an XForms instance embedded into HTML. XForms' Model-View-Controller (MVC) approach takes the XML structure (such as a document instance of a CMDI profile) as its data model. The view part of the MVC approach transforms the XML document into HTML/CSS. All control elements, in particular for saving the document's content, are also provided. The forms are defined in a declarative manner.

The described metadata editor is generated using a CMDI profile and its corresponding configuration file. The XForms technology then allows users to edit metadata files with standard browser technologies. When the users complete fields of the form in the browser, they fill in a CMDI document instance which serves as the data model in XForms' MVC approach. XForms technology generates a metadata file, which is saved in the CMDI file repository on the server (see Figure 1). This can be, for instance, the primary data repository containing the metadata file for descriptive purposes.

4.5 XForms + XHTML Processing

As most web browsers do not support XForms, it is required to interpret the generated XForms by other means. The transformation into browser-interpretable HTML/JavaScript is performed either on the client-side using a library such as XSLTForms (see `http://www.agencexml.com/xsltforms`) or on the server-side using a Java library on top of Tomcat (such as *betterFORM*, see `http://www.betterform.de/`). Both

libraries are available as open source (licensed under GPL, BSD/Apache, respectively).
The resulting XHTML with JavaScript is interpretable by browsers, displaying forms as standard web forms. For the user, the underlying XML is not visible, but the XML structures are represented in the form. Upon saving the file, the same libraries process the input to modify the original data model, i.e. the CMDI file that was being edited.
The XForms+XHTML processing module is both a web application and a web service: the web application being able to communicate with the delivering process on the server is used for editing. The web service, however, can be deployed as a viewer for the metadata, especially with read-only access to the underlying data.

4.6 Current Implementation Status

The current prototypical implementation of ProFormA rests upon an XSLT-based generation of XForms on the server-side using the betterFORM library in a Tomcat environment with the described input. Fig. 2 exemplarily shows Mozilla Firefox executing one of the generated XForms for TüBa-D/Z, the Tübingen Treebank of Written German (see `http://www.sfs.uni-tuebingen.de/en/tuebadz.shtml`).
Once the user has submitted all data, the instantiated data model (the CMDI document instance valid with respect to the CMDI profile) is written by the web application into the specified directory. This way the CMDI-XML-file can be edited without exposing the user to the XML code.
As a minimum, the editor currently supports those CMDI profiles defined within the NaLiDa project[1]. The reason for this is the focusing on the project's user groups that led to the creation of the editor in the first place. As the NaLiDa profiles have exactly been developed for the needs of these groups, the current ProFormA implementation also starts with a support of these profiles. However, it is possible to integrate other CMDI profiles than those of the NaLiDa project. Further investigations need to be made estimating the effort which would occur due to the structure's diversity of different profiles in CMDI.
As one point of entry to the suite, users are presented with a picklist of supported profiles, where they can select the type of resource they want to describe. Based on the name of the resource, an empty CMDI file is created and stored.
The resulting editor is a standard form as can be found on many websites and also resembles questionnaires or paper-type forms. In XForms fields have *labels* (the name of the field), *hints* for providing quick reference, *help* for definitions as well as *alerts* for warnings and errors in the case of invalid input. The current implementation uses the help-function to refer to the respective entry of the data category in ISOcat.

[1]Until March 2012, 10 CMDI profiles have been developed within the NaLiDa project for different resource types, such as corpora, lexicons, experimental data or tools. So far, these profiles are only accessible in the private section of the Component Registry, as otherwise no modifications would be possible in the present development cycle of the CMDI metadata creation process.

Each form resembles an overview document for a resource description. In cases where closed vocabularies are available, a selection list is implemented instead of a string input. For extremely long selection lists, such as language identification codes (7679 codes in use) or country identifiers (246 codes available), a 'semi-open' selection list was chosen for usability purposes. For instance, in the language case a couple of expected languages are provided as a picklist, but if the desired language is not among them, users can insert another language. This is a standard type of selection list in XForms.
In terms of validation, an external validator is being used. The external validation helps to avoid performance bottlenecks, for example, because long lists of restricted vocabularies would slow down a built-in validation process significantly.
In summary, the basic requirements are fulfilled in the current implementation status. Enhancements are expected on the user interfaces' presentation styles, also speeding up the generation process. Finding enhanced default values and grouping strategies for converting the CCSL into XForms+XHTML will also result in a changed layout and improved usability.

5 Conclusion and Future Work

Given that the entire sustainability infrastructure is built around resource creators, all its components, including the ProFormA metadata editor presented in this paper, need to be easy to use. Completing forms is an appropriate way of supporting non-expert users to create metadata, especially if these forms provide context-dependent help for most of the form elements. The form layout is defined in declarative configuration files specific to given CMDI profiles. XForms technology gives the (web-based) editor for free once an XForms specification file has been defined.
In the context of future work, the extended requirements will be addressed next (cf. Section 3). This mainly includes improvements in the integration of definitions of both data categories via ISOcat and components via the Component Registry. Thereby, users will get additional support in the metadata creation process, contributing to the metadata's quality by avoiding, for instance, tag abuse.
An additional task is to investigate the editor's capability of supporting other CMDI profiles than those developed in the NaLiDa project. First tests show that other profiles can be processed the same way, but some of the resulting forms are rather complex. Thus, an evaluation of the arising amount of work needs to be conducted on representative samples.
Another aspect concerns the robustness of the XForms generation which needs to be tested. Enhancements would also profit from the full range of elements that XForms offers. For example, when the CMDI profile asks for a date, most XForms interpreting libraries provide a calendar chooser, which generates a form element accepting only valid dates as input. The use of these additional features of XForms requires the correct assignments of data types in the profiles, which is currently sometimes simplified. In later versions, the configuration files will provide the data types, as extracted from the profiles. The data types and *display attributes* will then be used in the XForms generation.

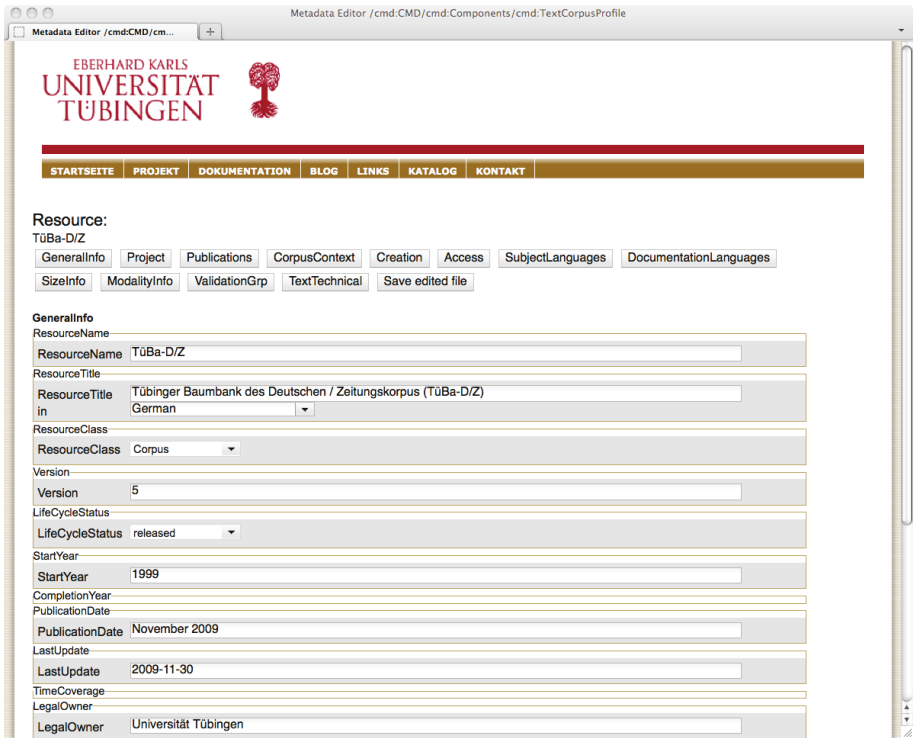

Figure 2: Metadata editor displaying a part of the metadata for the resource TüBaD/Z, a treebank for German.

Additionally, the editor is intended to access data available from the Linked Data Initiative. When users enter the names of persons, institutions or locations, autocompletion should be used to access information stored in databases such as the authority files of the German National Library (see `http://www.dnb.de`) or geographical databases (such as `http://www.geonames.org`). This also involves linking to material made available by the Linked Open Data Community (see `http://linkeddata.org/`) in the future.

Another prospective development is the integration of the editor into the workflow of archiving resources in a primary research data repository. As the components of the ProFormA suite are available as services, all prerequisites for integration are fulfilled (see (Dima et al., 2012)).

6 Acknowledgements

The editor presented in this paper includes components and services developed by different projects. Their implementation was made possible by funding within the CLARIN-D (Common Language Resources and Technology Infrastructure, `http://clarin-d.net/index.php/en/`) project of the BMBF (Federal Ministry of Education and Research, `http://www.bmbf.de/en`), the SFB 833 (Collaborative Research Center 833: The Construction of Meaning - the Dynamics and Adaptivity of Linguistic Structures, `http://www.sfb833.uni-tuebingen.de/`) funding by the DFG (German Research Foundation, `http://www.dfg.de/en`) and the NaLiDa project (Sustainability of Linguistic Data, `http://www.sfs.uni-tuebingen.de/nalida/en/`) grant by the DFG in the LIS program (Scientific Library Services and Information Systems, `http://www.dfg.de/en/research_funding/programmes/infrastructure/lis/index.html`).

7 References

R. Barkey, E. Hinrichs, C. Hoppermann, T. Trippel, and C. Zinn. 2011a. Komponenten-basierte Metadatenschemata und Facetten-basierte Suche: Ein flexibler und universeller Ansatz. In J. Griesbaum, T. Mandl, and C. Womser-Hacker, editors, *Information und Wissen: global, sozial und frei? Proceedings des 12. Internationalen Symposiums für Informationswissenschaft (ISI 2011)*, pages 62 – 73, Boizenburg. Hochschulverband für Informationswissenschaft, Verlag Werner Hülsbusch.

R. Barkey, E. Hinrichs, C. Hoppermann, T. Trippel, and C. Zinn. 2011b. Trailblazing through forests of resources in linguistics. In *Digital Humanities*, Stanford: Stanford University, June.

J. M. Boyer. 2009. XForms 1.1. W3C Recommendation 20 October 2009, See `http://www.w3.org/TR/2009/REC-xforms-20091020/`.

D. Broeder, M. Kemps-Snijders, D. Van Uytvanck, M. Windhouwer, P. Withers, P. Wittenburg, and C. Zinn. 2010. A data category registry- and component-based metadata framework. In *Proceedings of the 7th Conference on International Language Resources and Evaluation (LREC 2010)*, Malta.

K. Coyle and T. Baker. 2009. Guidelines for Dublin Core Application Profiles. Dublin Core Metadata Initiative. Technical report, Dublin Core Metatdata Initiative, 05. `http://dublincore.org/documents/2009/05/18/profile-guidelines/`.

E. Dima, V. Henrich, E. Hinrichs, M. Hinrichs, C. Hoppermann, T. Trippel, T. Zastrow, and C. Zinn. 2012. A repository for the sustainable management of research data. In *Proceedings of the 8th Conference on International Language Resources and Evaluation (LREC 2012)*, Istanbul.

IMDI. 2001. Metadata elements for session descriptions, draft proposal version 2.5, June. `http://www.mpi.nl/world/ISLE/documents/draft/ISLE_MetaData_2.5.pdf`.

ISO 12620:2009. 2009. Terminology and other language and content resources – specification of data categories and management of a data category registry for language resource. Technical report, ISO.

G. Simons and S. Bird. 2008. OLAC metadata. Technical report, Open Language Archive Community. `http://www.language-archives.org/OLAC/metadata-20080531.html`.

H. Thompson, D. Beech, M. Maloney, and N. Mendelsohn. 2004. XML Schema Part 1: Structures. Second edition. `http://www.w3.org/TR/xmlschema-1/`. W3C Recommendation 28 October 2004.