Component Metadata Infrastructure Best Practices for CLARIN

Thomas Eckart
Leipzig University
teckart@informatik.uni-leipzig.de

Twan Goosen
CLARIN ERIC
twan@clarin.eu

Susanne Haaf
BBAW
haaf@bbaw.de

Hanna Hedeland
Hamburg University
hanna.hedeland@uni-hamburg.de

Oddrun Ohren
National Library of Norway
oddrun.ohren@nb.no

Dieter Van Uytvanck
CLARIN ERIC
dieter@clarin.eu

Menzo Windhouwer
CLARIN ERIC/Meertens Institute
menzo.windhouwer@meertens.knaw.nl

Abstract

This paper introduces the CLARIN Component Metadata Infrastructure (CMDI) Best Practices guide as currently developed by the CMDI and Metadata Curation Task Forces. This guide, combined with the technical specification of CMDI 1.2, will become a useful resource in creating high quality metadata records that function well in the CLARIN infrastructure. Thus, the scope of the best practices goes beyond purely schematic constraints and ranges from modelling guidelines, e.g. "prefer elements over attributes", to rather low-level technical guidelines, e.g. "use the UTF-8 encoding for your records", and includes steps to incorporate in workflows, e.g. "use the CLARIN Curation Module to assess the quality of records". This paper actively solicits feedback on a public draft version (CE-2017-1076) for the first official version of the guide.

1 Introduction

Last year, 2016, saw the release of both version 1.2 of the Component Metadata (CMD) Infrastructure (CMDI) [CLARIN ERIC 2017] and a first complete technical specification [CMDI Task Force 2016]. This new version provides new possibilities, which are gradually opened up by the ecosystem of tools and registries in CMDI. One of the key properties of CMDI is its flexibility, which makes it possible to create metadata records closely tailored to the requirements of resources and tools/services. However, design and implementation choices made at various stages in the CMD lifecycle might influence how well or easily a CMD record is processed and its associated resources are made available in the CLARIN infrastructure. So far, knowledge on this has been scattered around in various documents, web pages, and even tacitly in experts' minds. To make this knowledge explicit, the CMDI and Metadata Curation Task Forces have teamed up to create a Best Practices guide. This guide, together with the technical CMDI 1.2 specification, will become a valuable knowledge base and will help (technical) CMDI users to bring their CMD records to their full potential use within CLARIN. By August 2017, the writing of this guide is still an ongoing effort, but to foster discussion and feedback, a draft version of the guide is published as CE-2017-1076 [CMDI and Metadata Curation Task Forces 2017]. The next sections outline the guide, describe its scope and give glimpses into its current content.

2 Scope

The target audiences for the Best Practices guide include 1) CMD modellers who select or create components and profiles to describe certain language resources, 2) developers who create metadata converters, forms, or editors to create records that comply with these profiles, and 3) CMD creators who create records by hand. End users of the infrastructure are, in general, not directly exposed to CMDI, but if so should be fully guided by an application (with its own user guide) and shielded from the

technical details. Hence, this group is not a target audience of this guide. Furthermore, using this Best Practices guide will require a basic understanding of CMDI – it is not intended as a tutorial.

In general the CLARIN infrastructure is able to deal with any valid CMD record that is based on a profile from the CLARIN Component Registry. The technical specification [CMDI Task Force 2016] acts as a baseline to specify valid CMD profiles, components, and records. The scope of the best practices goes beyond purely schema-related constraints and ranges from modelling guidelines (e.g. "prefer elements over attributes"), to rather low-level technical guidelines, e.g. ("use the UTF-8 encoding for your records"). As a consequence, the CLARIN infrastructure will be able to extract information more efficiently from CMD records that meet the best practices and, hence, provide the end users with improved access to those records, the context they originate from and the resources they represent.

3 Outline

The first two sections of the CMDI Best Practices guide follow the lifecycle of Component Metadata: 1) modelling CMD profiles and components and 2) authoring CMD records. Best practices are provided for the various CMD constructs encountered at each stage. Both these sections also provide guidelines regarding the workflow, e.g. (sequence of) actions to take and tools to use, and indicators of potential problems (also known as 'smells'). Some approaches and problems, which can be considered 'crosscutting concerns', having aspects that are limited to neither modelling nor authoring, are covered in a separate section. The next sections provide insights in the current content of the guide.

4 CMDI Best Practices for CLARIN

4.1 Modelling Component Metadata

CMD modelling takes place on two similar, but distinct levels: the CMD profile and the CMD component level. The former is the higher of the two levels and represents the content scope of a full CMD record. Profiles and components are composed of (other) components, elements and attributes. Components represent aspects of metadata descriptions and serve as building blocks. This distinguishes components from profiles, which only serve as blueprints for metadata records. This gives rise to two sets of modelling best practices, in addition to a general modelling principle that we will cover first. This modelling principle relates to components, profiles, and concepts, advising to make these "as generic as possible and as specific as needed". Rather than a concrete guideline, this best practice can be considered a 'commandment' from which other more specific best practices follow. It reflects the principle of reuse, around which CMD has been designed, and the requirement for metadata modellers to always consider the potential of reuse even when modelling for a specific project or domain. This applies to, for example, naming of elements, scope of components, and the cardinality of attributes.

4.1.1 Component Modelling Best Practices

Detailed documentation of the right type should be provided in the right place. Here, descriptive documentation should be distinguished from semantic identification (which is best done by means of a concept link), and operational instructions. Best practices related to naming serve to promote clarity and consistency. According to these, component, element, and attribute names should be in English, verbose, avoiding abbreviations and acronyms, and not unnecessarily specific. Modellers should aim for a uniform naming 'pattern' (e.g. usage of upper and lower case) across their components, although consistency may be compromised by reusing existing components. Content validity and quality strongly depend on defining the right constraints and value schemes. The modeller should discourage usage of empty string values but rather demand values of a specific type where possible, selected from a vocabulary if feasible, even for elements of a binary nature. Elements are generally more suitable than attributes, and the latter should only be used to model value annotation. In practice, reusability is often hampered by over-specific modelling. The Best Practices guide provides suggestions for modelling components with broad reusability in mind. It urges modellers to reuse existing components wherever possible, or as a fall-back solution attempt to 'recycle', i.e. base new components on existing ones.

4.1.2 Profile Modelling Best Practices

Since modelling at the profile level methodologically overlaps with modelling at the component level, the component best practices also apply to profile modelling as far as documentation, naming,

constraints, value schemes, and the use of vocabularies go. Following the general reuse principle, profile creators are advised to compose their profiles from existing components as much as possible. Each of a profile's components should deal with a distinct aspect of a resource, avoiding semantic overlaps between different parts of a profile. The modeller should ensure that metadata can be created for broad user groups, which implies modelling for verbose, self-contained resource descriptions that cover a minimal set of concepts, e.g. license and language, as agreed upon by the CLARIN community.[1] Providing proper documentation is at least as important for profiles as it is for components, as metadata creators will primarily be exposed to CMD definitions at the profile level, starting with the profile selection process.

4.1.3 Workflow

The workflow sections provide information on CMD related workflows and the tools and services currently available to support them (e.g. the SMC Browser, the CLARIN Curation Module, and available CMDI Validators[2]). For CMD modelling, the aspects covered include assessing usage and overall quality of profiles, components, or concepts and handling lifecycle management, versioning, and visibility.

4.1.4 Problem Indicators ('Smells')

So-called 'smells' are indicators of bad or inefficient design of components or profiles and are a concept originating from design analysis in computer programming. Discovering such an indicator does not necessarily mean that a design is poor, but, instead, that there are warning signs that should be checked. An analysis of the public CMD components and profiles in the CLARIN Component Registry by the authors revealed several problem indicators.

As an example of such an indicator, a 'Standalone Profile' reuses no or very few components, which may suggest insufficient evaluation of the existing component stock in the modelling phase. 'Speculative Generality' is seen as a design flaw when a component or profile allows describing aspects of a resource that most likely will never occur in any real-world CMD record. In those cases a reduction of size and complexity may lead to simpler designs and improved intelligibility for users. Another kind of 'Speculative Generality' holds for a profile that is modelled to describe different types of resources, leading to disjoint instantiation patterns in CMD records. In those cases it may be useful to extract resource type independent aspects to a single component and create a profile for every resource type while reusing the common parts.

4.2 Authoring Component Metadata Records

A CMD record consists of two main sections: 1) an envelope, which contains (technical) metadata on the record itself and on the resources it describes, and 2) the component section, which should be a valid instance of a CMD profile.

4.2.1 Authoring the Envelope

An important best practice for the envelope is to include a reference to at least one resource. Otherwise, it remains unclear which language resources are described by the metadata record. The exact type of these resources should also be made explicit, so that generic tools, e.g. the Language Resource Switchboard [Zinn 2016], can process these resources correctly. Some resources consist of distinct parts, e.g. items in a collection or annotations and annotated text in separate files. Such decomposition should be reflected in the CMD file by listing each part as an individual resource proxy. Moreover, if the listed resources are related to each other in significant ways, those relationships should be represented as instances of Resource Relation, each of which is referencing the related resources as well as a concept describing its meaning. The technical metadata of a CMD record should also contain the identifier of the CMD profile used.

4.2.2 Authoring the Component Section

The component section must be a valid instance of the profile the record pertains to, i.e. it should assign values to all mandatory components, elements, and attributes as a minimum. In general, it should be

[1] These concepts generally correspond to the fields and facets of the Virtual Language Observatory
[2] Links to these tools and service can be found at https://www.clarin.eu/cmdi

aimed at providing 'complete' metadata information with regard to the selected profile, i.e. the information asked for by the profile's components, elements etc. should be provided as exhaustively as possible. This applies especially to covering the minimal set of concepts, as agreed upon by the CLARIN community, as those have great impact on the visibility of the record and its resources within the CLARIN infrastructure. In addition, components dealing with the sources of a resource (e.g. for a digitized text this might be a physical volume as found in a library) should be attended to, as well as distribution and access conditions. The information provided should be specific rather than generic (e.g. a list of two modality elements with values "gesture" and "speech" is preferred over a single modality element with value "multimodal").

4.2.3 Workflow

For the creation of CMD records, the focus of the workflow section is on how to integrate quality checks, which often relate to best practices, into individual workflows. This also includes considering the visualization of the metadata in the VLO, which is a result of both the mapping of concepts to facets and the normalization of values.

4.2.4 Problem Indicators ('Smells')

'Smells' can also be identified for metadata records. These include indicators on a variety of levels. As an example, an extensive record size (several megabytes or more) may indicate an inappropriate granularity, where, instead of a single resource, a complete resource collection is described. In these cases it may be helpful to restructure the outsized file to a collection record referring to several individual records, each describing a single resource. Other aspects indicating 'fitness' of a record are sufficient coverage of the minimal set of concepts, as agreed upon by the CLARIN community, as well as the used level of detail: a record can only be accurately represented in central applications – like the VLO – if elements containing the precise name or a helpful description of the resource are instantiated. The complete guideline contains many more hints, similarly derived from identified shortcomings in real-life CMD records.

4.3 Common Approaches and Problems

In addition to best practices for modelling and authoring CMD, the final version of the guidelines will contain sections covering several common approaches and problems such as multilingual metadata, hierarchies and granularity, licensing, dealing with uncertainty and modelling for metadata conversion. Next to these topics, some common use cases will also be discussed, for example how to create a CMD profile and/or records for a common resource type, e.g. WordNet.

5 Conclusions and Future Work

The CMDI Best Practices guide is a currently ongoing, joint effort by the CMDI and Metadata Curation Task Forces. While the guide itself remains work in progress, the best practices shortly outlined in this paper already illustrate how it can help CMD modellers and authors making decisions that enable the CLARIN infrastructure to give end users optimal access to the CMD records, and thus to the associated language resources provided by the CLARIN centres. The authors, and the task forces they represent, very much welcome feedback on the draft version of the guide [CMDI and Metadata Curation Task Forces 2017], so the final version will meet its aim to help the CLARIN community, improve the infrastructure and the access to language resources in general.

Acknowledgements

The authors would like to thank the members of both the CMDI and Metadata Curation Task Forces of the Standing Committee of CLARIN Technical Centres for their continuous and valuable feedback on the (draft) CMDI Best Practices guide.

References

[CLARIN ERIC 2017] CLARIN ERIC. 2017. *CMDI 1.2*, https://www.clarin.eu/cmdi1.2.

[CMDI Task Force 2016] CMDI Task Force. 2016. *CMDI 1.2 specification*. CE-2016-0880.

[CMDI and Metadata Curation Task Forces 2017] CMDI and Metadata Curation Task Forces. 2017. *CLARIN's CMDI Best Practices Guide*. CE-2017-1076. (draft)

[Zinn 2016] Claus Zinn. 2016. *LR Switchboard (software)*. CE-2016-0881.