Challenges in Modelling a Richly Annotated Diachronic Corpus of German

Stefanie Dipper[1], Lukas Faulstich[2], Ulf Leser[2], Anke Lüdeling[1]

[1]Institut für deutsche Sprache und Linguistik
{Stefanie.Dipper,Anke.Luedeling}@rz.hu-berlin.de
[2]Institut für Informatik
{faulstic,leser}@informatik.hu-berlin.de

Humboldt-Universität zu Berlin
Unter den Linden 6, D-10099 Berlin

Abstract

This paper presents the design and architecture of a diachronic corpus of German. We describe the corpus architecture with a focus on the use and restrictions of XML as the data exchange and storage format. In our approach, a relational database will supplement the XML representation to support sophisticated search and presentation facilities. This is a report about ongoing work; the architecture presented here is being developed in a pilot study.

1. Introduction

This paper describes the design and architecture of a diachronic corpus of German with texts from the 9th century (Old High German) to the present (Modern German). This corpus will be built by the large-scale Germany-wide project Deutsch.Diachron.Digital (henceforth DDD).[1]

We describe the corpus architecture of DDD with a focus on the use and restrictions of XML as the data exchange and storage format. We argue that a corpus based on a collection of XML files is not sufficient to support sophisticated search and presentation and that therefore a relational database with an information retrieval extension serves our needs better. We plan to use a graph-based representation for the corpus and to provide powerful import/export methods to support various XML-based formats.

Historical texts are of interst to scholars in many fields (historical linguistics, theoretical linguistics, philology, history, philosophy, ...). However, although many historical texts (manuscrips and early prints) have been digitized in a number of projects (for example, TITUS[2], Bibliotheca Augustana[3], MHDBDB[4]; for an overview, see Kroymann et al., 2004), the historical corpus situation for German is not satisfying: There are no common standards for digitization (this pertains to the question of the best source—manuscript or edition—as well as to the level of diplomaticity and the quality of collation), header information, or annotation on any level. Projects often do not conform to existing standards such as TEI (Sperberg-McQueen and Burnard, 2001) or XCES (Ide et al., 2000). There are no common search interfaces. Many of the digitized texts are not available to a wider audience.

As a reaction to this, DDD aims at creating a generally available, unified resource with common standards and search programs for scholars in the above mentioned fields as well as for interested laypeople. The architecture needs to be highly flexible to cover all the requirements.

The paper is structured as follows. We first present the DDD project and its requirements. We then describe related corpus projects and, finally, give a description of the implementation concept, addressing the general architecture (the data model), import and export methods, and the XML-based representation (the exchange format).

2. Corpus Architecture

The architecture of the DDD corpus has to satisfy different types of requirements: (i) requirements specific to diachronic corpora, (ii) requirements due to the heterogeneity of the corpus, and (iii) requirements due to different types of users. These requirements call for a flexible corpus architecture on the one hand, and for maximal standardization in digitization and annotation on the other hand.

2.1. Requirements for Diachronic Corpora

Our corpus is a historical corpus in that it deals with older texts; moreover, it is a diachronic corpus because it comprises texts from different language periods. Both properties come with requirements that differ from the requirements of corpora consisting of texts from one language period only.

Multi-modality For many historical linguistic research questions, it is necessary to refer to the manuscript facsimiles. Therefore, some parts of the corpus will be aligned by page or line to manuscript facsimiles.

For teaching purposes it is sometimes instructive to hear older texts read out. Hence, certain texts will be aligned to sound files.

[1]The project developed from a Germany-wide initiative (at the moment 15 universities are involved) and is in its beginning phase, with the final funding decision still pending. The architecture presented here is being developed in a pilot study within the Forschungsverbund Linguistik-Bioinformatik, financed by the Senatsverwaltung für Wissenschaft, Forschung und Kultur, Berlin. See `http://korpling.german.hu-berlin.de/ddd/`. Due to previous work of the project partners, DDD can start out with a considerable amount of digitized texts, which are partially annotated, by varying types of information.

[2]`http://titus.uni-frankfurt.de/indexe.htm`

[3]`http://www.fh-augsburg.de/~harsch/augusta.html`

[4]`http://mhdbdb.sbg.ac.at:8000/index.html`

Integration of external resources The corpus will be linked to external resources, like, e.g., the electronically available lexicons for Middle High German[5].

Multi-linguality, alignment The corpus is multi-lingual. First, although there is a continuous development from Old High German to Modern German, there are enough differences between the periods to speak of several languages here. Second, there are many texts, especially in the Old High German period, that contain Latin parts. Some texts are direct (interlinear) translations of Latin, others are interpretations of Latin texts. The interlinear translations are especially interesting for research on word order: any difference in word order between the original Latin text and the German translation points to strong constraints of the Old High German grammar. This means that we need a word-to-word alignment (more precisely: an alignment of n to m words) between Old High German and Latin in these texts, cf. the example in Figure 1, taken from *Tatian* α 2,7 (Sievers, 1961).

Besides text-internal alignments as in the above example, alignments between different texts will be made as well. Examples are alignments between different manuscript versions of the same story (e.g. the various manuscripts recounting the Nibelungenlied), between different editions of the same manuscript, or between a manuscript and its edition.

Another type of example is the alignment of corresponding words of different periods. The purpose of such alignments is to trace the changes a word undergoes in the course of time. For instance, *imbizs* (Old High German) corresponds to *inbizze* (Middle High German), which finally evolved into *Imbiss* in Modern German.

The alignment will be encoded by means of 'hyper lemmas'. A hyper lemma is a set comprising the (normalized) lemmas of different periods that correspond to each other. The lemmas then are linked to the actual words occuring in the text.[6]

Structural annotation The layout of old texts may bear important linguistic information. For instance, words are often split in two parts by line breaks, and it is an open research question how often the location of such breaks coincides with syllable or morpheme boundaries.

Therefore, the texts in the corpus will be structurally annotated, both graphically (marking lines, pages, etc.) and logically (verses, sentences, etc.). Note that this leads to conflicting annotation hierarchies.

Smallest reference unit The token=graphemic word-based encoding of modern corpora is not directly applicable to historical texts.

Historical texts make heavy use of abbreviations, e.g. the character sequence *er* is often replaced by a ˜, as in *d*˜ (= *der* 'the'). Such abbreviations will be spelt out in the normalized, unabbreviated word form. Ideally the normalization allows for a reconstruction of the abbreviation sign ˜ and the corresponding, spelt-out characters.

h	a i	z	a	n		(Alemanic, diplomatic)
h	ê	z	a	n		(Middle Franconian, diplomatic)
h	e i	z	a	n		(Normalization)

Figure 2: Character alignment of dialectal and normalized form

A further example is provided by orthographic variations as they occur in different dialects (which may indicate differences in phonetics and/or phonology). For instance, *ai* in the Alemanic dialect usually corresponds to *ê* in the Middle Franconian dialect. The normalized form (which abstracts away from dialectal variation) uses *ei* to encode this sound. The characters corresponding to each other should be aligned, as sketched in Figure 2.

To model these requirements appropriately, the smallest units of reference in the corpus representation must be single characters. Further possible applications of such a character-based annotation include the encoding of initials and ligatures (paleography), linebreaks, and alliteration.

This has the additional advantage that morpheme boundaries can be annotated and the differences between graphemic and lexical word can be easily marked.

Meta-annotation The annotation of historical texts is at best semi-automatic. This means that often several annotators work on the same text. It is useful to keep record of the annotation task by means of meta-annotations. Meta-annotations refer to other annotations and encode information such as the annotator of the referenced annotation, the date of annotating, or the tool applied in the annotation task. Comments can be added to any annotation unit in the same way.

2.2. Requirements Due to Corpus Heterogeneity

The DDD corpus is heterogeneous with regard to the depth of annotation and its composition.

Annotation depth Depending on the research question, the requirements with regard to annotation of a corpus differ. While for many linguistic questions, rich annotation is desirable, there are philological and lexicographic questions where corpus size may be more important than annotation depth. To satisfy both requirements as best as possible, the depth and type of annotation will differ within the DDD corpus. This must be accounted for by the corpus architecture which should allow the user to select homogeneously annotated sub-corpora as a basis for further research.

The corpus will be composed of three subcorpora, which we call the extension corpus, the core corpus, and the presentation corpus (cf. Figure 3). The extension corpus consists of texts that are annotated only with structural information (see below) and header information (based on the standards TEI (Sperberg-McQueen and Burnard, 2001) and XCES[7] (Ide et al., 2000)), encoding bibliographic information. In addition, the core corpus will be annotated

[5] `http://gaer27.uni-trier.de/MWV-online/MWV-online.html`

[6] The question of which lemmas of different periods correspond to each other is a difficult one, involving aspects of semantics (word meaning), morphology, etc. The DDD project aims at defining clear criteria for hyper lemmas.

[7] `http://www.xml-ces.org/`

Et	*non*	*erat*	*illis*	*filius,*	*eo quod*	*esset*	*Elisabeth*	*sterilis*	(Latin)
inti	*ni*	*uuard*	*in*	*sun,*	*bithiu uuanta*	*Elisabeth*	*uuas*	*unberenti*	(OHG)
and	not	was	them	son	because	Elisabeth	was	sterile	(Gloss of OHG)

'and they did not get a son because Elisabeth was infertile'

Figure 1: Word-to-word alignment of a multi-lingual source text (Latin – Old High German), Tatian

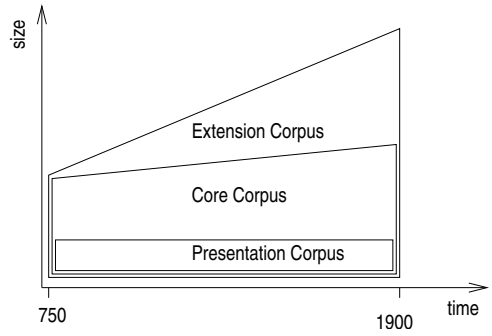

Figure 3: Composition of the planned DDD corpus

with (normalized) lemma information, part-of-speech tags and inflectional morphology.[8] Other annotation levels, e.g. information structure or syntax, can be added to texts from either subcorpus. Some texts—the presentation corpus—will be aligned to manuscript facsimiles or sound files. All types of annotation will be based on existing standards, if available (e.g., STTS (Schiller et al., 1999) for part-of-speech tagging, TIGER (Brants et al., 2002) for syntax annotation), which, of course, will have to be adapted to the special requirements of historical and diachronic data.

DDD intergrates a lot of already digitized material[9], which has to be brought to a common quality standard and annotated.

Corpus composition The corpus composition differs for the different language periods. The older language periods (Old High German, Old Saxon) can be digitized almost completely, while in the newer periods the corpus needs to be balanced with respect to a number of parameters like region, genre, dialect, etc.

Additional texts will be added in the course of the project. Hence, the corpus architecture must allow the addition of new texts. At the same time, it must be possible to identify reference corpora for each period.

2.3. User Requirements

The DDD corpus addresses scholars in many fields, e.g., linguists, lexicographers, philologists, historians. The needs of these user groups differ with respect to (i) search facilities, (ii) the presentation of the corpus, and (iii) export options.

[8]The annotation of historical texts heavily depends on manual work. In this paper, we do not address the issue of how the information will be annotated.

[9]This material was digitized in different projects at partner universities and, among others, includes (parts of) the TITUS corpus, the Bonner Mittelhochdeutsch corpus, and the Digital Middle High German Text Archive.

Search facilities Lexicographers search for the use and collocations of a word or word form. In a diachronic corpus, they can also look at meaning change or form change. For instance, at about 900 AD the word *imbizs* meant 'delicious meal', whereas the corresponding present-day form *Imbiss* means 'snack'. For lexicographic purposes, we therefore need full-text searches and collocation extraction.

In contrast, linguists often search for annotated information such as morphology, part of speech, syntax, e.g. to investigate the change of word order in German. This usually involves complex, cross-level queries.

We are convinced that the requirements of the prospective user groups cannot be satisfied by providing a single search interface. Therefore, we envisage the provision of at least two levels of searching: one simple full-text search including only the digitized text, and another interface providing access to the full annotation.

Corpus presentation Similarly, the ideal visual presentation of the corpus depend on the type of user. The texts will be represented with or without annotation or with selected annotation types only. In addition to a Web interface, presentation with external viewers (e.g., PDF) should be supported.

Export options The corpus (and search results) will be represented by a primary XML exchange format. This allows the user to further process and manipulate the data by external tools.

An XML format will also be used as the exchange format for annotated texts within the project. However, external editors and annotation tools may require or produce documents in different formats.

Standards compliance Finally, existing linguistic and IT standards should be applied wherever possible to facilitate access to the corpus (including future access), to ease the application of external tools, to make data reuse possible, and to allow for comparison and exchange with other corpora.

3. Related Corpus Projects

DDD is inspired by research on historical corpora, multi-lingual corpora, and multi-modal corpora. We first give a broad overview before going into more technical detail.

Historical corpora DDD cannot be modelled directly after existing historical corpora of other languages because most of them are smaller and made with a specific purpose in mind (literary goals or linguistic goals, but not both). For example, the diachronic part of the Helsinki Corpus (roughly 1 million words), which was originally collected for research on variation and language change

(Rissanen et al., 1993), is now annotated linguistically with part-of-speech information and syntax[10]. Other historical corpora that have more annotation levels encompass only a certain language period (like the Lancaster Newsbook corpus, which contains 17th century newsbooks[11]) or are even more specific (compare the corpus of the Nibelungenlied[12]). However, even though the overall corpus architecture cannot be directly copied, existing historical corpora are the basis for many decisions concerning the different annotation layers.

Multi-lingual corpora DDD shares many of the problems of multi-lingual corpora, in that we need alignment between texts and also within the same text. As mentioned above, some of the texts are direct interlinear translations from, e.g., Latin to Old High German. Here we need word-to-word alignment within the same text.

In many cases we have different manuscripts of the same text (as the manuscripts A–C of the Nibelungenlied)—these need to be aligned as well. The problem goes further: in order to track lexical change, all of the texts in the corpus need a common 'normalized' lemma layer (the 'hyper lemma' annotation).

Multi-modal corpora DDD can be modelled on multi-modal corpora, which have the task of connecting different representations of the same utterance—for example, a spoken sentence with its transliteration and the gestures that were made while speaking—with each other and with annotation layers.[13] Each representation and annotation layer is represented in a different file, resulting in a multi-layer stand-off annotation. Roughly spoken, all files are connected via reference to a common base line (or time line for speech data).

The data model for DDD (which is presented in detail in Sec. 4.2.) is inspired by this architecture, but generalizes it by permitting multiple time lines for the same text: a diplomatic rendering of the original text serves as a base line for the graphemic view of the text (volumes, pages, lines, graphemic words), whereas the logical view of the text (chapters, sections, paragraphs, sentences, lexical words) refers to the time line of a normalized version of the text. Both time lines are aligned by annotations that link graphemic with lexical words (cf. Figure 6). Each further representation or annotation layer (normalization, part-of-speech tags, structural inormation, etc.) can refer to either one of these base lines or to annotations within other layers. In XML, each annotation layer can be stored in a separate file which uses XPointer URLs to refer to a base line. In this way, we can deal with conflicting hierarchies, different modes of representation (text, graphics, speech) as well as with the fact that not all texts in the corpus are annotated in the same depth.

[10] `http://www.ling.upenn.edu/mideng/`

[11] `http://www.ling.lancs.ac.uk/newsbooks/`

[12] `http://www.blb-karlsruhe.de/blb/blbhtml/nib/uebersicht.html`

[13] For examples of multi-modal corpora, see the SmartKom corpus (`http://www.smartkom.org/`) or the GeM corpus (`http://www.fb10.uni-bremen.de/anglistik/langpro/projects/gem/newframe.html`); see also below.

4. Implementation Concept

To repeat the above requirements: the DDD corpus is multilingual, multi-modal, and has to support different and varying annotation levels. The smallest unit of annotation is the character and DDD has to support conflicting hierarchies. The corpus must be searchable with intuitive and, at the same time, powerful search tools that can search on all annotated levels.

We first present the overall architecture of DDD in Sec. 4.1., where it is specified that the DDD corpus is stored in a central database. The data model that will serve as the basis for structuring the corpus within the database is introduced in Sec. 4.2. We then focus on how texts can be exported from / imported into the database (Sec. 4.3.). In Sec. 4.4., we present the XML formats that will be used for exchange with other project partners and for publication of the corpus.

4.1. System Architecture

We propose a Web-based system architecture where users search or browse the DDD Web server via standard Web browsers (cf. Figure 4). For digitization and annotation, the project partners can download XML files using a Web browser, apply external tools to these files, and upload the modified XML files again to the DDD Web server. External tools may also communicate directly with Web services offered by the DDD Web server. For instance, a lemmatization tool might access a lexicon at the DDD server via a Web service.

The Web server routes user requests to a module of the application logic tier, which in turn communicates with the relational database system storing the corpus. The application logic comprises several search interfaces, import and export converters, and administrative modules for access control, diagnosis, etc.

The corpus itself is stored in a relational database system containing a full-text retrieval component. Compared to the storage of a corpus in a flat file, this yields several advantages:

- Sophisticated search facilities on text, header data, and annotations: full-text search can be combined with search criteria on header data; complex conditions on annotations and information referenced by annotations can be formulated; etc.
- Extensive support for statistical analysis in modern SQL: SQL:1999 and SQL:2003 (Türker, 2003) incorporate several statistical operators developed for data warehousing applications, which can be used for analyzing large sets of annotations.
- More natural representation of non-hierarchical data (cf. Sec. 4.2.): in XML, non-hierarchical relationships must be expressed using ID-references, which have to be handled by special means.
- Independence from document formats: in Sec. 4.3. we show that various import and export formats can be

supported without imposing the restrictions of a particular format to the database.

- Multi-user capabilities: relational database systems can support a large number of concurrent users. In particular, they are able to successfully handle conflicts arising from concurrent write operations.
- Robustness, scalability, maturity: modern database systems provide excellent means for recovery and backup. They can be easily extended to cope with increasing demands for storage and throughput.
- Longevity: by using industry standards such as SQL, the chance to ensure long-term operation of the DDD corpus is increased.

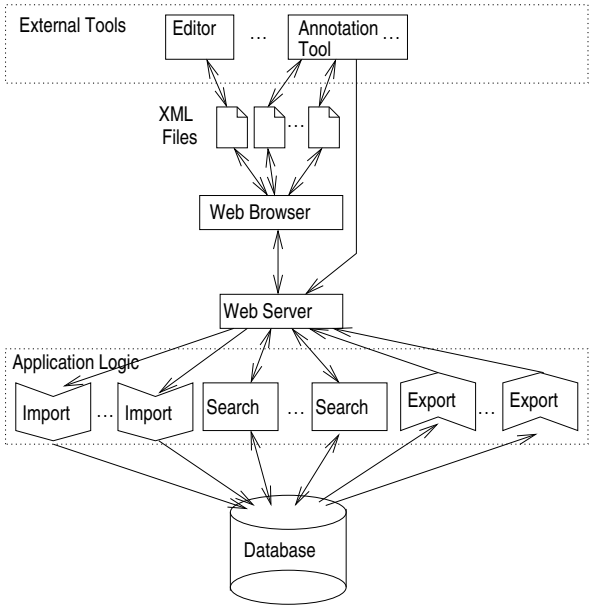

Figure 4: System Architecture of DDD

4.2. Data Model

Although the DDD corpus will be stored in a relational database, we restrict ourselves to a specialized data model for annotated texts rather than using the relational data model in its full generality.

There are two popular data models for multi-modal corpora: the annotation graph (AG) model (Bird and Liberman, 2001) and ordered directed acyclic graphs (ODAGs), such as the NITE object model (NOM) (Carletta et al., 2003). Annotation graphs model annotations as arcs that connect time points on the time axis of a signal. Annotation graphs can be stored easily in relational databases and searched efficiently by translating queries into SQL. However, the AG model has some shortcomings. For instance, parent-child relationships cannot be represented in AGs without extending the data model with special child/parent arcs (Teich et al., 2001). Without this extension, the dominance relation between a non-branching node and its only child is not encoded. Meta-annotations or alignments cannot be represented directly but need to be expressed by introducing equivalence classes (i.e., annotations are linked by assigning them identical attribute values).

The ODAG-based NOM does not share these limitations. Annotations are represented by nodes. Annotation values are stored in form of node attributes. The domination relation between nodes is modeled explicitly by parent-child relationships. Each node may refer to a span of the underlying text. In this case, the child nodes must refer to non-overlapping text spans contained in the span of their parent node. The order of child nodes must correspond to the order of their spans in the underlying text.

We have two requirements which go beyond the NITE model: (i) we want to represent the whole corpus within the same data structure to enable cross-references between texts, and (ii) we want to permit complex annotation values, which cannot be represented as node attributes, a need that has been recognized also in (Brugman and Wittenburg, 2001).

For DDD, we propose a data model based on ODAGs that is presented in Figure 5. Two prominent features of our model are:

- A collection of texts is associated with each ODAG. This collection comprises the source texts of the corpus and, in addition, notes, comments, and other free-text annotations. Every node may reference a span in some text associated with the ODAG. This generalizes the NOM where all texts (“signals”) are synchronized and references cannot point to a specific text. As in NOM, the span referenced by a node must be contained in the spans referenced by its ancestor nodes. Moreover, the spans referenced by the children of a node must be disjoint and the textual order of the spans must be consistent with the order of the child nodes.
- Annotations with complex values are seen as relationships between ODAG nodes. A complex annotation is a node with a child that marks a region of the source text, and one or more children representing (facets of) the annotation value. Alignments are annotation nodes having several children referring to the source text(s).

It is straightforward to use this data model as a generic database schema. However, this approach lacks efficiency. We plan to investigate the efficiency of object-relational features offered by SQL:1999 and SQL:2003. These features can be used to organize nodes by name into a hierarchy of tables that store each node together with its attributes. Parent-child relationships have to be stored in bridge tables since – differing from XML document trees – they are of cardinality $m : n$.

This approach has the advantage that we can represent the whole corpus as a single ODAG under a single root node. Each annotated text is represented by a subgraph that is rooted in a child of the corpus root node. In Figure 6, the structure of a prototype ODAG for the DDD corpus is sketched.

4.3. Import/Export Methods

For presentation, exchange, and support of existing tools, an XML representation of the ODAG is necessary.

An XML document can be modeled as an ordered tree, which is a special case of an ODAG. However, the ODAG

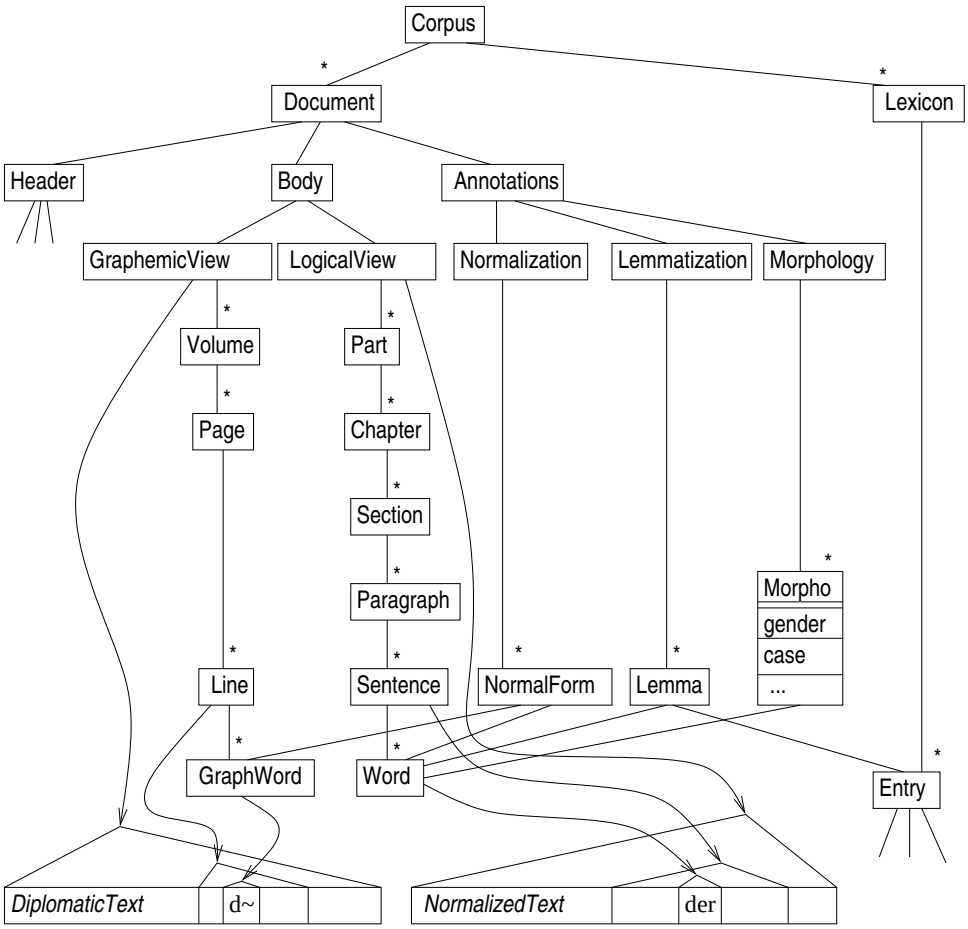

Figure 6: Prototype of DDD corpus schema

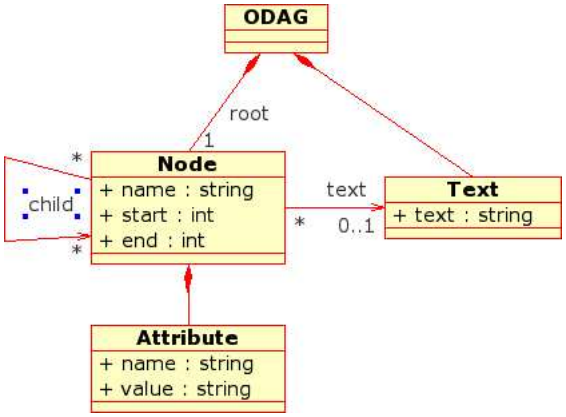

Figure 5: UML model of the DDD datamodel

stored in the database is in general not a tree. Hence, to export an XML document from the database, a tree has to be generated on the basis of the stored ODAG. Moreover, we may want to include only certain annotation layers in the XML document or would like to use names for document elements and attributes that differ from the names used in the database. When importing XML documents into the database, an inverse transformation has to be performed.

Hence, a powerful and flexible method for transforming a source ODAG into a target ODAG is needed to support the import and export of XML documents into/from the database. Requirements for this transformation method are:

Projection: Only certain annotation layers may be needed in the target ODAG. For instance, one might want to present only the physical structure of a document on a Web page.

Selection: The source ODAG may be restricted to a certain part of a text. For instance, a Web page might present only a single chapter of a text. Conversly, an XML document created by an external tool may contain tool-internal data that is to be excluded from import into the database.

Folding and Rearrangements: The target ODAG may contain different transformations of the same part of the source ODAG. For instance, we may want to generate an HTML document that presents each text line as a table whose rows correspond to different annotation layers. This requires a different transformation of the same line for each annotation layer.

Derived Attributes and Elements: Complex annotation values may be derived from separate simple attributes. For instance, the complex STTS part-of-speech value *VAINF* ('verb, auxiliary, infinitive' (Schiller et al., 1999)), as used, e.g., in the TIGER corpus (Brants et al., 2002), may be derived from the

atomic part-of-speech value *verb* and the specifications *type=auxiliary* and *inflection=infinite*.

Interchange between Element Content and Attributes: Values of annotations that are stored as attributes within the database may be represented as element content in an XML document and vice versa.

Context-Sensitivity: Nodes may be transformed in different ways, depending on the context. For instance, a word node may be copied verbatim in the context of a sentence, while in the context of a grammatical annotation, the word node is transformed into an XPointer reference.

Addressing: IDs or XPointer URLs identifying document elements or addressing text regions need to be generated (on export) and resolved (on import) to support stand-off annotation formats.

Encoding/Decoding of Conflicting Hierarchies: Although our primary exchange format avoids the problem of conflicting hierarchies by using separate annotation files, we need to support other formats to represent several hierarchies within the same document. Several solutions for this problem have been proposed by the TEI. From these, we plan to support at least milestones, fragmentation, and virtual joins:

Milestones: creating milestones means replacing subsequent elements (e.g., pages) by empty milestone elements (e.g., page breaks). Decoding of milestones means reconstructing annotations spanning the regions separated by the milestones.

Fragmentation: annotations with a lower priority must be split into several parts at the borders of higher-priority annotations. On import, these annotation fragments must be merged again into single annotations.

Virtual Joins: virtual joins are based on fragmentation but have additional IDREF attributes linking the fragments.

4.3.1. Generic Mapping

To apply existing transformation technology for XML, we use a generic mapping between ODAGs and XML document trees that replicates all shared nodes in the ODAG. Node IDs are generated and stored in an extra noderef attribute to keep track of the original nodes. Text referenced by a node is inserted as PCDATA, interleaved with the document elements representing the children of the node. In addition, the span of the referenced text is described by the attributes text (URI to text), start, end (span).

To facilitate the creation of XML documents for import into the database, redundant content need not to be reproduced. Document elements sharing the same noderef attribute are unified into a shared ODAG node. Empty document elements with a noderef attribute are treated as node references. However, all non-empty elements referring to the same node must have the same content. The unmarked text of the XML document is extracted, concatenated, and an appropriate reference to a span of this text is added to each node unless the node refers explicitly to a text span using the text, start, end attributes. Nodes with such explicit span references may omit the referenced text content, in order to reduce redundancy.

4.3.2. XML Transformation

The XML document resulting from the generic ODAG-to-XML mapping can be transformed in various target formats using general purpose transformation methods like XSLT[14] or STX (Streaming Transformations for XML[15]). XSLT is quite expressive and satisfies most of our requirements in a natural way.

The encoding of conflicting hierarchies by XSLT is not straight-forward, but can be implemented using the timing attributes to select and clip elements of a subordinate hierarchy (see Figure 7). However, this is quite inefficient.

4.3.3. Need for a High-Level Transformation Language

Both methods for encoding conflicting hierarchies result in quite complex and verbose stylesheets that are hard to write manually. This problem could be solved by developing a more high-level transformation language.

Moreover, the ODAG stored in the database can become arbitrarily large. To make the XSLT-based approach scalable, only a subgraph of the database ODAG containing the information to be included in the target document should be exported. An XPath expression could be used to select a node set. The selected subgraph would then be formed by all nodes reachable from this node together with a new synthetic root node. Further specification options might be useful to control the replication of shared nodes depending on the path used to reach them.

Hence it would make sense to define a new transformation language that is better suited to our requirements. This language might be implemented either by generating XSLT stylesheets with additional parameters controlling the generic transformation or by a specialized transformation mechanism of its own.

4.4. Exchange Format

We distinguish two XML-based exchange formats. The first one will be used within the project as the exchange format for annotated texts. This format is the result of the generic mapping described in Sec. 4.3.1., which maps ODAGs stored in the database to XML document trees. This redundant representation separates conflicting hierarchies and the various annotation layers. It uses node and span references to keep track of shared nodes and the alignment of annotations with the underlying texts. For import, redundant content may be omitted.

The second format represents the 'external' exchange format. This format will serve as the official exchange format, wich is made available to the research community. It will be XCES-compliant—which means that, with the current version of XCES[16], not all encodings of the DDD corpus can be represented adequately.

[14] `http://www.w3.org/Style/XSL/`
[15] `http://stx.sourceforge.net/`
[16] `http://www.xml-ces.org/`

5. Conclusion

In this paper, we presented the architecture for a large-scale, diachronic, multi-modal corpus for German. We first sketched the diverse requirements for digitization and annotation that result from the type of data, the different user groups, and their research questions.

In a pilot study, we developed a flexible corpus architecture to answer these requirements. The corpus will be represented by an ODAG and stored in a relational database. An XML-based representation will be derived from the ODAG representation, which serves as the exchange format within the project. In addition, an XCES-compliant XML representation will be made available for research purposes.

In our architecture, the smallest units of reference are characters. There are two time lines: first, the diplomatic text, focusing on physical properties of the source text; second, the normalized text, focusing on logical properties. The corresponding annotations often result in conflicting hierarchies. To find a suitable representation and efficient methods of manipulation for these hierarchies will be a major point in our future work.

6. References

Bird, Steven and Mark Liberman, 2001. A formal framework for linguistic annotation. *Speech Communication*, 33(1,2):23–60. `http://arxiv.org/abs/cs/0010033`.

Brants, Sabine, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith, 2002. The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories, Sozopol*.

Brugman, Hennie and Peter Wittenburg, 2001. The application of annotation models for the construction of databases and tools: Overview and analysis of MPI work since 1994. In *IRCS Workshop on Linguistic Databases*. `http://www.ldc.upenn.edu/annotation/database/papers/Brugman_Wittenburg/20.2.brugman.pdf`.

Carletta, Jean, Jonathan Kilgour, Timothy O'Donnell, Stefan Evert, and Holger Voormann, 2003. The NITE object model library for handling structured linguistic annotation on multimodal data sets. In *Proceedings of the EACL Workshop on Language Technology and the Semantic Web (3rd Workshop on NLP and XML, NLPXML-2003)*.

Ide, Nancy, Patrice Bonhomme, and Laurent Romary, 2000. XCES: An XML-based standard for linguistic corpora. In *Proceedings of the Second Language Resources and Evaluation Conference (LREC)*.

Kroymann, Emil, Sebastian Thiebes, Anke Lüdeling, and Ulf Leser, 2004. Übersicht über diachrone Korpora. Technical report, Institut für Informatik, Humboldt-Universität zu Berlin. `www.linguistik.hu-berlin.de/ddd/publikation/HistorischeKorpora.pdf`.

Rissanen, Matti, Merja Kytö, and Minna Palander-Collin (eds.), 1993. *Early English in the Computer Age*. Mouton de Gruyter.

Schiller, Anne, Simone Teufel, Christine Stöckert, and Christine Thielen, 1999. Guidelines für das Tagging deutscher Textcorpora mit STTS. Kleines und großes Tagset. Universitäten Stuttgart and Tübingen, `http://www.ims.uni-stuttgart.de/projekte/corplex/TagSets/stts-1999.pdf`.

Sievers, Eduard (ed.), 1961. *Tatian. Lateinisch und altdeutsch mit ausführlichem Glossar*. Paderborn: Schöningh, 2nd edition.

Sperberg-McQueen, C. M. and Lou Burnard (eds.), 2001. *TEI P4: The XML Version of the TEI Guidelines*, chapter 31: Multiple Hierarchies. Text Encoding Initiative. `http://www.tei-c.org.uk/P4X/NH.html`.

Teich, Elke, Silvia Hansen, and Peter Fankhauser, 2001. Representing and querying multi-layer corpora. In *Proceedings of the IRCS Workshop on Linguistic Databases*. University of Pennsylvania, Philadelphia.

Türker, Can, 2003. *SQL:1999 & SQL:2003 - Objektrelationales SQL, SQLJ & SQL/XML*. Heidelberg: dpunkt.verlag.

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <!-- ... -->
  <xsl:template match="line">
    <xsl:variable name="clipStart" select="number(@start)"/>
    <xsl:variable name="clipEnd" select="number(@end)"/>
    <line no="{@no}">
      <xsl:apply-templates select="ancestor::document/structure/logical/part/verse[
(number(@start) &lt; $clipEnd) and (number(@end) &gt; $clipStart)]">
        <xsl:with-param name="text" select="$text"/>
        <xsl:with-param name="clipStart" select="$clipStart"/>
        <xsl:with-param name="clipEnd" select="$clipEnd"/>
      </xsl:apply-templates>
    </line>
  </xsl:template>

  <xsl:template match="verse">
    <xsl:variable name="start">
      <xsl:choose>
        <xsl:when test="number(@start) &lt; $clipStart">
          <xsl:value-of select="$clipStart"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:value-of select="number(@start)"/>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:variable>
    <xsl:variable name="end">
      <xsl:choose>
        <xsl:when test="number(@end) &gt; $clipEnd">
          <xsl:value-of select="$clipEnd"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:value-of select="number(@end)"/>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:variable>
    <verse no="{@no}">
      <xsl:value-of select="substring($text,$start + 1, $end - $start)"/>
    </verse>
  </xsl:template>
</xsl:stylesheet>
```

Figure 7: Detail of an example XSLT stylesheet that implements the fragmentation of verses within lines, using the timing attributes start and end. In each line element all overlapping verse elements are included and clipped at the borders $clipStart and $clipEnd of the line element. The text of the (clipped) verse element is computed as a substring of the unmarked text stored in $text.