universität wien

# DIPLOMARBEIT

Titel der Diplomarbeit

„Design and Implementation of a Research Infrastructure for a Corpus of Spoken ELF”

Verfasser

Stefan Majewski

angestrebter akademischer Grad

Magister der Philosophie (Mag. phil.)

Wien, 2011

| | |
|---|---|
| Studienkennzahl lt. Studienblatt: | A 343 |
| Studienrichtung lt. Studienblatt: | Anglistik und Amerikanistik |
| Betreuerin: | Univ.-Prof. Mag. Dr. Barbara Seidlhofer |

# Preface

I have been in the fortunate situation that I had the opportunity to work for a real scientific project as an undergraduate student. It is an interesting story how that came about: I've always been very curious to learn new things from apparently very distinct areas. Coming from an engineering background, I've been determined to try my fortune in a somewhat different area, the humanities. My technical background, combined with my interest in the humanities, got me into contact with the, at that time, just forming team of the Vienna-Oxford International Corpus of English (VOICE). Most certainly, the fact that I allow for a certain degree of serendipity was involved in establishing this link and getting me involved with the VOICE project.

From the first meeting with the team – i.e. the project director Barbara Seidlhofer and the two young researchers Marie-Luise Pitzl and Angelika Breiteneder – I had the strong feeling that this would be an exciting project with the appealing perspective to build a useful and real scientific resource. The prospect that this resource could facilitate research on English as a lingua franca and the open-minded atmosphere within the group got me involved and I joined the team as one of its initial members.

The project was driven by the spirit to explore a new area. An area that has been neglected the worth to be researched, so far. My part in the project touched many aspects from bringing my perspective into the conceptual work of the group to the implementation of the required research tools. Chores like keeping the basic IT infrastructure of the project alive, were also part of my work.
I developed programs that assist the transcribers (e.g. VoiceScribe), implemented a project database and designed the transformation from the VOICE transcription system to an XML corpus format. In the course of the first three years, we managed to successfully apply for a follow-up grant that allowed us to further improve the corpus resource and that provided enough time to design and develop VOICE Online.

It is good to have this thesis completed not too long after the publication of VOICE XML, as it documents many conceptual aspects of the annotation and the data-format.

## Typographic Conventions

This text uses a variety of technical terminology to express concepts as well as implementation details. Nonetheless, there is significant overlap between the terminologies, as related concepts are expressed in similar terms for different purposes, contexts and meanings. For example, the meaning of a feature annotated in VOICE is described in prose according to its meaning, its formal place in the VOICE transcription format and its rendition in the XML corpus format. To make the meaning and the interrelation of terminologies more accessible, the expressions of related concepts are most appropriately identified by labels indicative of their meaning. This means, in many cases, by identical labels. Therefore, it is necessary to introduce a typographic distinction to ensure that the text is unambiguous and readable at the same time. To keep the terminology sufficiently separate and the text readable, the following typographical conventions are used throughout the text.

- `annotation` for components of VOICE's corpus annotation in an abstract sense. This font is used in cases where the notion of annotation matches the components that are present in VOICE transcripts. The names are chosen to match their conceptual correspondent terms. In the discussion of the use of some particular annotation, as for example when describing its conceptual constituents, the `annotation` rendering is only used when referring to a particular annotation and not the concept.
- `<element>` for XML elements. These are used for the description of XML elements used for the corpus encoding of VOICE's annotation system.
- `@attribute` for XML attributes. These are, similar to XML elements, used to describe attributes used for the corpus encoding of VOICE's annotation system.

In most cases it is sufficient for the reader to keep two simple rules in mind. First of all, most of the time, understanding the concept the label relates to is enough to understand the text. The labels, though, are described at the appropriate places in the text. Secondly, three types of interrelated terminology are set apart from the text in a monospaced typeface: annotation, just indicated by the different font, XML elements enclosed in pointy brackets (i.e. <>) and XML attributes prepended with an AT-symbol (i.e. @).

# Contents

# 1 Introduction

Many research projects in the humanities today have to face methodological challenges concerning the technological implementation of their research infrastructure. The Vienna-Oxford International Corpus of English (VOICE) (cf. VOICE Project 2009) encountered such challenges and some of them are probably typical for humanities research projects. They range from incorporating legacy data from pilot projects to bringing together upfront technological standards and humanities' research methods to create versatile and sustainable digital resources. While in practice it is a huge task to master either of these areas on their own, the true challenge lies in bridging and reconciling methodologies without conflict. That is, bringing together methodologies from different disciplines in the service to be of use for achieving the original goal. Interdisciplinarity, therefore, is in this case not the ultimate goal in itself, but a means to achieve a goal by building on the experience of other disciplines. Especially when dealing with structured data, such as textual data, it is useful to use knowledge from the computer sciences. One can draw from decades of experience in this area. Nevertheless, one has to be careful to choose wisely from the options offered by other disciplines. The fundamental problem is that if methodologies do not match, conflicting methodologies can spoil the result. The issue is that

> [...] different disciplines do their abstraction in different ways, focus on different aspects of experienced reality, follow different conventions of enquiry. Interdisciplinary relationships are established where areas of possible convergence can be identified, where one discipline accommodates the concepts or procedures of another, thus changing the scope and manner of its abstraction. (Widdowson 2006: 95)

Bearing in mind this potential change of abstraction, a humanities research project can benefit from actively searching for possible methodological overlap with other disciplines. The important question that has to be asked, and more importantly answered, is the question whether the methodology is still suitable for the targeted research purposes. For text corpora, this convergence lies to a certain extent in storing linguistic data in data structures that can be efficiently processed. Obviously, when using methodologies from other disciplines to organise the data, such as methods developed within the computer sciences, it is still an ultimate requirement to be capable of answering linguistic questions. That is, to keep these questions in scope of the abstraction imposed by the chosen methodologies. Taking care of this requirement, a humanities research project can increase the possibilities and therefore the benefit from the work that is put into the creation of a resource.

The declared goal of the VOICE Project has been to create and publish a resource that allows for the description and investigation into the nature of English as it is used as a lingua franca (cf. Seidlhofer 2001). With this objective, the project is located in the area of descriptive and applied linguistics. English linguistics, if at all concerned with the description of non-native speakers' English, has until recently primarily focused on

learner's English. Learner corpora proved a valuable resource to find differences between learners' English and native speaker's English, such that teaching materials could be designed to help learners to approximate closer to native speaker norms. Corpora with this goal necessarily focus on the conceptualisation and the emphasis of errors and deficiencies. The perspective of the VOICE project is very different, as the purpose of the project is to create a resource that helps to research and understand aspects of English as it is used as a lingua franca. Hence, VOICE follows a rather different approach, as it collects naturally-occurring spoken English of speakers with various different lingua-cultural backgrounds. The speakers use English to achieve and communicate their own real-life communicative goals. An attempt in focusing on errors and deficiencies would be misleading, as the strategies and the success of the speakers is more in the focus of our research. Therefore, it was crucial for the data to be collected where self selected participation in communication takes place between speakers of different first language (L1) backgrounds.

This thesis aims to introduce and explain the design of the research infrastructure developed for VOICE and to provide information on some of the core decisions. The project director has recognised from the beginnings of the initial pilot projects for VOICE that "[t]he feasibility of such a project is basically a question of methods and consequently has much to do with technology" (Seidlhofer 2001: 145). This text provides therefore a documentation of the resource VOICE as well as a description of the processes and technological and methodological choices taken in the course of designing and implementing VOICE. The reader will be introduced to the range of conceptual as well as technological foundations of the project with the goal in mind to allow the user more thorough research on the data. To reach this aim, this thesis includes a description of the annotation system of VOICE and arches from VOICE's particular data format towards the concrete implementation and deployment strategies used in the VOICE project.

From a documentation perspective, this text relates to the technical implementation of VOICE, like the VOICE *Mark-up Conventions*[1] (VOICE Project 2007) relate to the manual transfer of the original audio recordings to custom orthographic transcripts. Therefore, this thesis describes the transfer of VOICE's custom transcription format to a standards-based corpus framework. Where necessary, general conceptual aspects, like for example annotation and the representation of temporality within the annotation systems, are discussed in more detail. The documentation of these aspects of VOICE is intended to aid the careful researcher in understanding and using the many possibilities which the corpus provides as a language resource.

The VOICE Project is a research project that has mainly been funded by two subsequent substantial research grants by the Austrian Science Fund. This fund was sufficient to set up

[1]The transcription of VOICE is defined in the VOICE *Mark-up Conventions* and *Spelling Conventions*. Transcription relates here always to the transliteration of language and not a phonetic, phonemic or other kind of transcription. Except for onomatopoeia, which are transcribed in phonemic IPA.

and finance a research team of four main researchers to accompany the research director[2]. This group was responsible for the corpus design and creation. Even though some of the transcription took place within the core team, it was necessary to hire and train freelance transcribers to be able to meet the target size of one million words. Thus, it was required to coordinate the effort and the participation in the workflow of a varying number of people beyond the core VOICE team.

The resources that were created in the corpus-creation workflow are the original audio recordings, meta-data, transcripts and the XML corpus files derived from this information (See *figure 1*). The conceptualisation of the resources and how they relate to each other

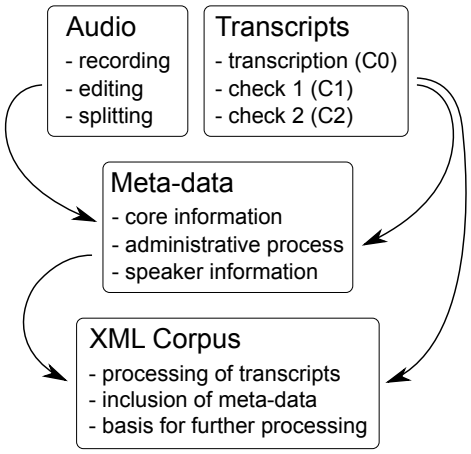



Figure 1: Resources and main steps of preparation involved in the compilation of VOICE. The arrows denote dependencies between resources. That means that a resource at the end of an arrow has to be created either after or dependent on the resource at the beginning of the arrow.

in the process of corpus building will be explained in the chapters on annotation (see *2. Annotation p.4*), transcription (see *4. Redesigned Transcription Format p.29*), the meta-data database (see *5. The VOICE Meta-data Database p.45*). and the corpus format (see *6. Corpus Publication Format p.55*) of VOICE.

As the work-sharing requires a fair amount of coordination and documentation, it is necessary to administer and track the work-flow. Therefore, it is required to define and evaluate the progress of the project to be able to define and adjust the target and process for the milestones on the way to the final product. VOICE's database served as a core infrastructure to maintain meta-data on the individual speech events, but also to keep track of the project's progress (See *5. The VOICE Meta-data Database p.45*).

[2]The persons centrally involved in the creation of VOICE are the founder of the VOICE project and the project director Barbara Seidlhofer and the four researchers Angelika Breiteneder, Theresa Klimpfinger, Marie-Luise Pitzl and myself. See *VOICE - Terms of Use* (VOICE Project 2009).

# 2 Annotation

When working with linguistic data, especially when analysing data, soon, the need to capture the result of the analysis emerges. In other words, the challenge is to find a way to keep and be able to re-use the findings. Furthermore, we usually want to indicate which part of the source the findings are derived from and include a description of what they are. As basic as it might seem, conceptually this is very similar to the scribbling of marginal notes when reading a scientific paper. Usually, when using the term 'annotation' a great emphasis is put on structure and consistency. The additional value supplied by annotation is the provision of previous analytic or interpretative material that complements the source material. This additional information can then be used as the foundation for further research.

Consider the following example taken from VOICE:

**S4:** <soft> thank you </soft> (2) {S1 searches for sheets (2)}
**S1:** <soft> [S5/last] </soft>
**S5:** <soft> mhm </soft>{S1 hands S5 some sheets}<soft> thank you </soft>
**S1:** okay e:r (.) <to S6> and yours is missing er we have to (.) <1> look after</1>wards. @@ </to S6> (.)
**S6:** <1> it's okay </1>

Source: *Vienna-Oxford International Corpus of English* (VOICE 2009: EDsve421.11-15)

Figure 2: This sample is taken from VOICE Online. The annotation is displayed in the default style of VOICE Online. That is, it is shown as mark-up directly within in the text.

Apart from the transcribed words, there is annotation that identifies speakers (e.g. S1), indicates events within the text (e.g. {S1 hands S5 some sheets}) or overlapping speech (e.g. <1>). At a basic level, this is all data based on some kind of analysis[3]. For now, the precise meaning of the annotation in the example is not important. Nevertheless, it is essential to see that annotation is in principle a very natural and simple concept in descriptive research. Most readers will at that point have gathered some of the fundamental properties of annotation. In the following, we will further explore this area, introduce the generic framework of the annotation graph (see *2.2. What is Annotation, Actually? p.7*) and develop requirements that annotation has to meet, such as theory neutrality (see *2.3. Theory Neutral Annotation p.12*). As VOICE is a corpus of spoken language we will explore how annotation can be used to express temporal aspects, such as sequential and simultaneous speech.

For the design of any text corpus, annotation is a crucial concept as it represents information on the resource on several different analytic levels. The most obvious level is, for a linguistic resource, the representation of linguistic features. Nonetheless, the explicit expression of the design criteria, such as for example the sampling criteria for a given

[3]For a more detailed account of methodological considerations see *VOICE Recording-Methodological Challenges in the Compilation of a Corpus of Spoken ELF* (Breiteneder, Pitzl, Majewski & Klimpfinger 2006) and *Textbasierte Transkription in VOICE* (Majewski forthc.).

resource, is also a central aspect. That means that annotation is not only applicable to specific linguistic data, but also to different levels of the resource.

Without any kind of annotation, a text corpus would only be a mere collection of data, where the rationale for important aspects, such as the target population, the data collection and analytic information, such as linguistic aspects, would remain opaque and useless to a researcher using the material. The targeted annotation is a thread that guides through the whole process from data-collection and transcription to the provision of a queryable resource.

Annotation is, when comparing contemporary corpora, omnipresent. For using the available corpora, it is in many cases sufficient to intuitively use the provided annotation with the assumption that the annotators took care to consistently annotate the language resource at hand. For VOICE, this is of course similar. Most users of VOICE will not have much difficulties in understanding the annotation used for VOICE, without having to resort to the formal definition of the annotation used.

The following section will identify general strategies in conceptualising annotation and its relation to the annotated source. It explains the strategies used to make VOICE's annotation explicit enough to be intuitively interpretable in the intended way and furthermore helps to understand the rationale behind the annotation to be able to work with the more difficult or rare cases.

As annotation maintains such a central position in the creation and implementation of a linguistic resource, the subsequent sections discuss fundamental conceptualisations that are important for the understanding of the implementation of VOICE. It starts with the more general question of what annotation is (see *2.1. Systematic Annotation (p.5)* and *2.2. What is Annotation, Actually? p.7*), discusses the epistemic requirement of theory neutrality (see *2.3. Theory Neutral Annotation p.12*) and the expression of temporality within the linguistic layers of annotation (see *2.4. Temporal Relation Between Source and Annotation p.15*), as well as the different layers of annotation from linguistic annotation of small portions of text to meta-data descriptive of the whole resource (see *2.5. Levels of Annotation p.22*).

## 2.1 Systematic Annotation

One of the common requirements for the description of spoken language on the basis of transcribed speech is to provide additional information to the transcribed words. This additional information is conventionally labelled annotation. Therefore annotation can be seen as everything beyond the mere transliteration of the spoken words. In this view of annotation, the transliteration of spoken language may not explicitly be included as part of the annotation. Yet, the border between transcribed words and annotation is not clear cut and it is an almost arbitrary decision where the threshold that distinguishes annotation from source data lies. In any case, it is important to have a systematic definition and a consistent use of annotation.

Annotation is generally perceived to be interpretative. Therefore, especially with spoken language, a definition of annotation might equally well comprise the transcribed words of a conversation as the transcription itself is an interpretative endeavour. However, no matter where the line between annotation and source might be drawn, annotation presents some additional information to some source. The term 'source' is deliberately neutral here. The source that a given set of annotation relates to is specific to the purpose of a resource and can and usually will differ among encoding projects. For some projects the source might be the transliterated text. Nevertheless, the annotation could also relate to a primary source such as video or audio recordings. All kinds of data, written, auditive or visual can serve as source data for analysis and annotation. Where the annotation relates to a textual source, the orthographic representation would not be part of the annotation. In this case, the orthographic representation is the referenced object structure, that is, the annotated source. But, when annotating audio recordings it could be a valid assumption that the annotation comprises the orthographic representation of the audio recording. In this case, the orthographic representation is itself already regarded to be additional information to the recording.

If systematic annotation can establish a relation to the different sources, the question is how it is possible to distinguish source and annotation and how to know to which source a given annotation relates. Consider the distinction between annotated audio source and an annotated transcribed text. The source can be identified by answering the following question: Which points are unambiguously and directly identified by the annotation. Is it a time in the audio file or is it a portion of text one reads? The answer to this question would determine whether the former, i.e. an audio source, or the latter, i.e. the transcribed words, constitute the source. In VOICE, annotation is defined by stretches or points in the transcribed speech. It is the point of reference that determines where the border between annotation and annotated source may be drawn. Therefore, even though VOICE is based on audio recordings, the annotation relates to the transcribed speech.

Where a transliteration is regarded to be the annotated source object, annotation beyond transliteration might be as simple as prepending a speaker's contribution with an indication of the person who utters the given portion of speech (e.g. "S3: " for the third speaker in the transcript). The complexity of annotation is only limited by the needs, resources and the requirements of a particular study or (corpus) project. Even the later addition of further annotation that is based on newer analysis is possible. The annotation that is used for example to identify a speaker is placed in the text without a direct reference to the audio source. Therefore, this kind of annotation has to be regarded as being an annotation of the textual source and only indirectly related to the recorded audio material. In practice, the definition of the threshold between source and annotation is not crucial. For most resources, the relation is immediately visible. Nevertheless, it is in any case important

to see that annotation is structurally independent of the referenced source and provides additional systematic information, separate from but related to the language captured.

## 2.2 What is Annotation, Actually?

This question appears to be one of the few rarely answered questions in corpus linguistics. A reason for this might be that the answer to the question of what annotation is appears at first sight to be straightforward. In the following, I will use annotation as it is defined in a seminal paper in which the authors propose a *formal framework for linguistic annotation* (Bird & Liberman 2001).

> "Linguistic annotation" covers any descriptive or analytic notations applied to raw language data. (Bird & Liberman 2001: 23)

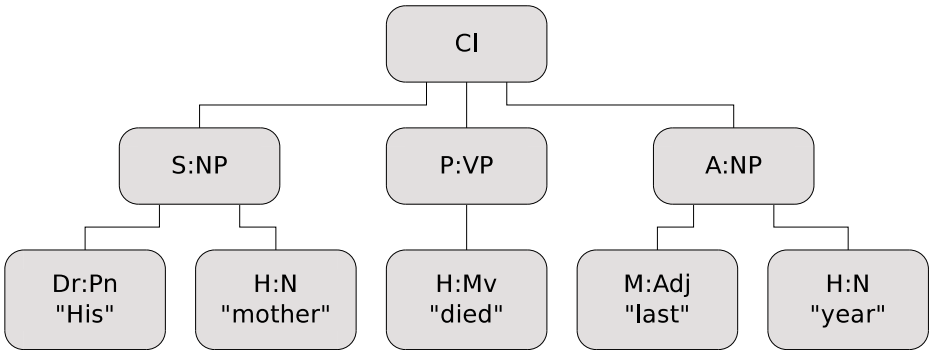
The definition is very broad, but it has two great advantages. First of all, it does not declare what "raw language data" (Bird & Liberman 2001: 23) is supposed to be. That means that it can be related to any kind of source material that might be relevant to a linguist. Therefore, it covers primary audio material as well as transcribed texts or even pre-existing text corpora. Consequently, language material of any kind can be enriched with annotation. Secondly, it covers any descriptive notation, irrespective of its structure. Hence, the focus is put on the general concept of annotation and not a particular concrete representation. Therefore, this definition avoids the problematic interchangable use of 'annotation' and 'mark-up', where 'mark-up' is generally understood to be something put into a text. Despite this, annotation can be realised as mark-up. There are many good examples for this as most contemporary corpora in linguistics follow this approach. Nevertheless, it is not as such a requirement that every annotation has to be realised as mark-up. The annotation could be put into a separate structure at a different place. It could, for example, be stored in a database alongside the original resource. Therefore, Bird and Liberman suggest the concept of an abstract "annotation graph".

> This provides a formal framework for constructing, maintaining and searching linguistic annotations, while remaining consistent with many alternative data structures and file formats. (Bird & Liberman 2001: 23)

The abstract representation of structural relations, such as relations that are represented by linguistic annotations, can be generalised with the concept of the 'graph'. The concept 'graph' was developed in the area of graph theory in discrete mathematics. Even though this may at first sound rather complex and certainly very formal, the concept is in practice very useful and even simple to understand. Most readers probably have seen and intuitively understood examples of an 'annotation graph' without relating it to its name proper. For this reason, it is best introduced and explained by taking an example from a domain that is likely to be familiar to the linguist reader. That is, it is an example from the domain of

linguistic taken from an introduction to English grammar where the concept of linguistic constituents is introduced at the example of a clause. Even though it requires still a small effort to understand the abstract concept behind these graphs, it is worth investing some time here. Understanding the basic principles behind an 'annotation graph' makes it, later on, much easier to work with the data and strengthens the confidence in claims that are possible on the basis of the annotated data. More complex structures, such as the VOICE annotations, are much easier to tackle with the analytic toolbox provided by this concept. Thus, in the following an annotation graph that is well known to most linguists is analysed and explained from the perspective of graph theory. This graph will be transformed step by step to the form that has been been proposed by Bird and Liberman (2001).

A graph like in *figure 3* will – since many textbooks and introductions to grammar include graphics similar to this – be familiar to most readers. It provides an analytical view on clauses, phrases and constituents. Nevertheless, this is only secondary here, as it is not the reason for choosing this example. The purpose of the example in *figure 3* is to be illustrative of annotation expressed as a graph. Its content is only in so far relevant, as it appeared to be convenient to take an example that is not from an utterly alien domain for linguists, but focus on the structure of the annotation.

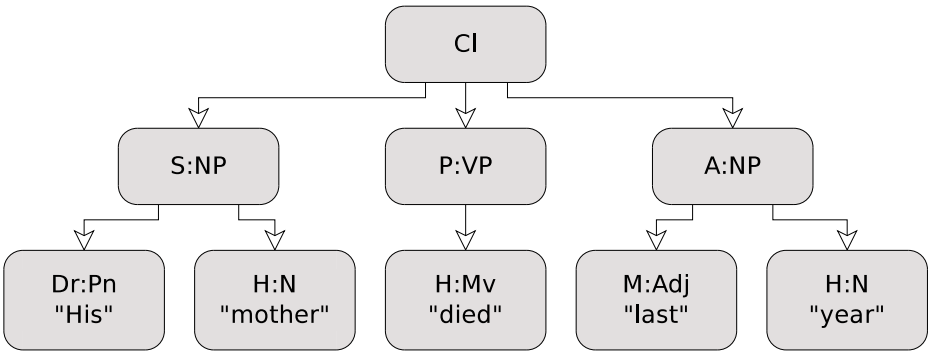



Source: Example based on *English grammar: an introduction* (Collins & Hollo 2000: 15)

Figure 3: Graph representing the structure of the sentence "His mother died last year" on the level of its constituents.

We can see in *figure 3* that there are two different types of items present in the graph. There are boxes and connecting lines between the boxes. First of all, the boxes represent linguistic entities. Secondly, the lines between boxes establish a relation between these entities. In our example the linguistic entities are constituents and the relations denote the composition of constituents from other 'subordinate' constituents. In graph theory, these two types are called 'vertex' or 'node' for the entities and 'arc' or 'edge' for their relation (cf. Bronshtein, Semendyayev, Musiol & Muehlig 2007: 348). In the following this text will adhere to the terms 'node' and 'edge' as these are more commonly used in informatics and appear to be slightly more illustrative of the concept. The fundamental concept that nodes are connected with edged arching from one edge to the other is in fact the foundation

of graph theory. Anything beyond this, including for example the layout of a representation on paper, is not part of the abstract graph.




Source: Example based on *English grammar: an introduction* (Collins & Hollo 2000: 15)

Figure 4: The graph from *figure 3* with the direction of the composition explicitly expressed by 'directed edges'

The edges in our example establish a relation we call 'composition'. That is, we see that a clause (Cl) is composed of a noun phrase (NP), a 'verb phrase' (VP) and a second NP. Even though the layout of the graph in *figure 3* indicates which node is composed of which, the line representing the edge of the graph does not show the direction of the composition, but only which nodes are related by means of a composition. That is, from a formal perspective both readings

- Cl is composed of NP
- NP is composed of Cl

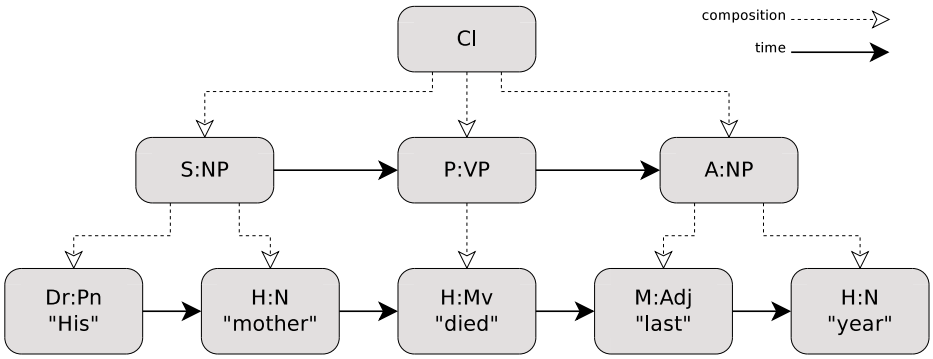
are possible. The second reading is, although formally possible, obviously nonsensical. We therefore conclude that relations in graphs have a direction. That is, they cannot be read in the same way from both sides. Hence, both readings

- Cl is composed of NP
- NP composes Cl

are equivalent and the expected and correct reading of the relation.

Even though the edges in the graph of *figure 3* appear to be 'directed', this is not shown as a property of the edges, but is implied by the overall layout of the figure. It is important to see that in this case, the implied property of the edges is established by convention. That is, the edges are arranged along the vertical axis of the figure. Nodes at the top are 'composed of' other the connected nodes below. The nodes at the bottom 'compose' the connected nodes at the level above. Apart from very simple graphs, it is often not possible to arrange the nodes, such that the direction of the relation is evident by the nodes' location in the figure. Nevertheless, it is important to remember the possibility of implied direction, as in practice many relations are established by convention in linguistic annotation (i.e. the

order of speaker contributions in VOICE). Formally, that is in graph theory, the direction is generally a property of the individual edge if the relation is directed. Its defining criteria are the 'initial point' and the 'terminal point'. The choice of nodes in this relation determines the direction of the edge. That is, it is directed from its 'initial point' to its 'terminal point'. Therefore, if an edge is directed, this is conventionally explicitly represented as part of the definition of the edge. That means that the initial point and the terminal point of the 'directed edge' are to be encoded so that the relation is clear. In a graphic representation this is normally achieved by using arrows to indicate the direction of a relation. The graph in *figure 4* is extended in this fashion. The arrow at one side of an edge points in the direction of the relation. The arrows in this example read from their root to their target as a 'composed of' relation. This enhanced version of the figure does not rely on the presupposition that the direction of the edges is clear by the top and bottom dichotomy.



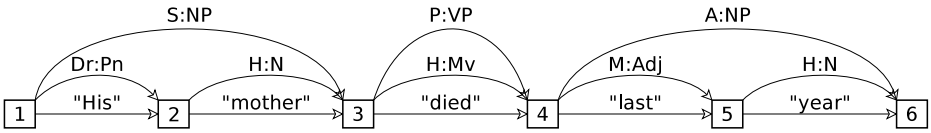

Source: Example based on *English grammar: an introduction* (Collins & Hollo 2000: 15)

Figure 5: The graph from *figure 4* with the order of the sequence explicitly shown by 'directed edges'.

Not only the direction of the edges is implied in *figure 3*, but also an omitted set of edges expressing the sequential order of the nodes. Every relation that is implied by any convention governing creation of the annotation and, hence, the interpretation of the graph can be expressed explicitly by introducing additional edges which is done in *figure 5*. While the graph in *figure 3* misses only the direction of the edges, the graph in *figure 4* nevertheless still misses a complete set of edges. The added edges in *figure 5* establish a 'followed by' relation between two nodes and make their temporal order explicit.

We have seen that graphs are a simple, yet effective means to express relations between entities. The entities are the 'nodes' of the graph and the relation are the 'edges' connecting the nodes. Furthermore, we have seen that graphs can have implicit relations if they are presented in a format following certain conventions. These relations, nevertheless, can be made explicit and are defined by the rules declared in the conventions that were used for the preparation of the data. This aspect is important, as in annotations of linguistic data many relations are established by convention (e.g. the order of speaker-contributions in VOICE). Nevertheless, all these annotations can be viewed as annotation graphs with their nodes

defining points in time and the relations annotating a span or point in time. This can be expressed by transforming the graph to a structurally different graph, where all annotation is referenced relating to the points between the annotated words. The graph in *figure 6* shows




Source: Example based on *English grammar: an introduction* (Collins & Hollo 2000: 15)

Figure 6: The graph from *figure 3*, where all nodes that are annotation are transformed to edges pointing to locations within the text.

the result of the transformation where the nodes of the graph refer to a point in time in the transcript and the directed edges connecting the nodes to the annotation. It is important to see that the linguistic entities from the initial figure are transformed from nodes to edges, as they are in fact annotations of a primary source (e.g. an audio recording or the transcribed words). The original graph from *figure 3* is a 'tree' which is a graph with the additional rule that any node can only have a single edge directed from another node towards it and an arbitrary number of edges directed from the node to other nodes. This way, a hierarchical type of graph structure is established which resembles a tree-like structure. The graph in *figure 6* resulting from the transformation on contrary is a flat linear structure where all edges are directed according to their sequential alignment. That is to say, no matter which type of annotation is of interest for analysis, it is always straightforward to establish a temporal order of the annotations. This is of special importance for the interpretation of the temporal order, which is crucial for spoken language corpora such as VOICE.

Nevertheless, there may be good reasons for choosing a different format for transcription. Even though we have seen that linguistic annotation can be transformed to form an abstract annotation graph, it is important to note

> that particular applications in the areas of creation, query and display of annotations may be most naturally organized in ways that motivate a user interface based on a different sort of data structure than the one [Bird and Liberman] are proposing (Bird & Liberman 2001: 43)

For this very reason VOICE features a transcription and corpus format that uses a text based mark-up, that is, mark-up that is directly written into the text. Eventough this textual mark-up does not, by its appearance, resemble a graph, this graph can be constructed from the structure and the conventions inherent to the transcription and corpus format. The structure of the VOICE transcription and corpus formats is thus not only determined by the structure of the annotation, but by what appeared to be a natural user interface for transcription and the technical constraints imposed by the corpus queries that are required to be possible with VOICE.

We can relate the possibility of an alternative representation of the annotation graph back to the example we discussed from *figure 3* to *figure 6*. Collins and Hollo (2000) propose a text-based hierarchical notation of the annotation called 'labelled bracketing' (cf. Collins & Hollo 2000: 15). The format of the alternative text-based representation is shown in *figure 7*. This is in fact an alternative text-based encoding of the annotation in the discussed figures, just as the VOICE transcription conventions are an alternative text-based encoding of the annotation for VOICE.

$$\begin{array}{ccccccccccccccc} & S & & Dr & & H & & P & & H & & A & & M & & H & \\ [ & ( & His & & mother) & & ( & & died) & & ( & & last & & year)] \\ & NP & & Pn & & N & & VP & & Mv & & NP & & Adj & & N & \end{array}$$

Source: Example based on *English grammar: an introduction* (Collins & Hollo 2000: 15)

Figure 7: Equivalent version of the annotation as in-text mark-up. Here, following the conventions of 'labelled bracketing' suggested by the authors of the example.

To conclude this discussion of annotation graphs, we have found that annotation can be expressed in different forms. One form is the directed annotation graph. Other forms can be tree-like structures or even text-based mark-up. All these types of mark-up can be transformed to the same form of a linear directed annotation graph, where all annotations are anchored to points in time or points in the transcript. With this simple abstraction, the temporal structure of the annotation becomes most directly and unambiguously accessible. This section has introduced formal aspects of annotation. The following section will continue with theory-neutrality, a fundamental qualitative and epistemic requirement that has to be met.

## 2.3 Theory Neutral Annotation

Annotation in linguistic corpora is used to document, analyse and understand phenomena that would be otherwise inaccessible to further linguistic analysis. It is important to see that this annotation already includes a significant portion of interpretation and analysis at the step of transcription. This systematic annotation is itself implicitly or explicitly governed by a theoretical framework that determines the decision where to put some particular annotation. Most researchers would agree that it is beneficial to keep corpus annotations ideally as theory-neutral as possible. In this respect, 'theory-neutral' means that the interpretative steps that led to the annotation are generally agreed upon and expected by the user of the resource, irrespective of the particular research question. This goal can obviously never be fully reached. This is due to several reasons. First of all, any kind of transcription and annotation is an abstraction and therefore a reduction of a given reality which in itself establishes a reality. Both realities should ideally have a reliable relation, that means the abstraction is in this case consistent throughout the data. Nonetheless,

it is important to remember that while abstraction mirrors a reality it is not possible to reconstruct the full nature of the primary reality. They cannot be identical, as otherwise there would not be any abstraction. They cannot fully match, neither in scope, nor in the consistency of the interrelation.

Any abstraction is therefore unavoidably to some extent a theory-governed *reduction* of some given reality. While an abstraction on the one hand makes it possible to increase the visibility of some selected aspects of the primary reality, on the other hand, the greatest part of this primary reality is irrevocably lost. This is a fundamental quality of representation – and hence abstraction – both beneficial and problematic. It is beneficial, because the loss of information increases the visibility and accessibility of the represented structures within the captured reality. It is problematic as the visible features are determined by the method of annotation and data processing. In the case of a linguistic corpus of spoken language, this first interpretative and reductionist step is the process of transcription. As the researchers involved in the creation of VOICE wanted to represent the language as faithfully as possible, this reduction can at points appear to be disappointing. "There is a sense in which any transcription of spoken text is inevitably indeterminate" (Burnard 2002). Although the process of transcription is inherently interpretative, the corpus builders have to make sure that the annotation is kept as theory neutral, as consistent and as close to the documented specification of the transcription methods as possible.

'Theory neutral annotation' refers to several distinct aspects of annotation. These aspects concern the methods employed, the structure of the annotation and the impact on studies that build on a given annotation set. Firstly and most importantly, the methods should be widely accepted by the scientific community. This means, secondly, that the general practice of supplied corpus annotation should follow fundamental, generally accepted and hence expected standards. Thus, the standards used should not be misleading but sufficiently explicit. Hence, any annotation that is new, innovative and leaving the trodden paths of corpus annotation should be especially well designed, documented, explicit and consistent. An example of an annotation concept unusual for spoken corpora is the use of the pronunciation variation and coinage (i.e. the annotation `pvc`, see *4.2.21. pvc p.40*)[4] in VOICE. This tag is therefore highlighted in the corpus documentation as well as in a dedicated publication on the methodology and possible inferences (cf. Pitzl, Breiteneder & Klimpfinger 2008). Any annotation that fails to follow these basic principles, that is, of being either theory neutral or explicit about the methodology used should therefore be handled with the greatest care. Thirdly, the methods of annotation should not interfere with the analysis to be carried out later. This last fundamental requirement is basically postulated to prevent hard-to-detect syllogisms in the reasoning of subsequent studies building on the annotated data. This means that any annotation-set created by using a methodology based

[4]The guidelines for transcription can be found in the appendix *VOICE Transcription Conventions (p.219)*

on theory $A$ in the process of annotation should not be used as empirical basis to support theory $A$ or a theory $B$ that is derived from theory $A$.

Such circularity would arise, for instance, if one were to examine a lexically transcribed and orthographically normalised corpus like VOICE for research that examines phonetic variation within words. The conclusion that only little variation occurs within the source data would be clearly wrong, but could be accidentally reached if a user of the corpus were not aware of the methodological decisions that determined the creation of the resource. To relate this to above stance on theory-neutrality, we summarise that the model used hides the feature, such that the influence of the methodology that has been used to create the resource invalidates any claims relating to a hypothesis that is dependent on this. Therefore it is not surprising that an expected phonetic variation is not observable in VOICE, even though it might be present in the original audio source material. It is hidden beyond the surface of the transcription and annotation system and therefore lost in abstraction.

The situation is even more difficult as the methodology used for transcription and annotation might not only hide some aspects beyond its immediate surface, but skew and distort the image of some specific features. Even though this appears to be negative, it is important to stress that the visibility of other features is increased with this reduction. In the example of VOICE the phonetic variation is hidden behind the lexical normalisation. Therefore, the occurrence and collocation of lexical items can be observed and accounted for more accurately with data based on a normalized lexical transcription system. This, again, points to the two important aspects of abstraction that are highlighted in the following. First of all, much information is bound to be lost. Secondly, and more importantly, some aspects of the described reality become more clearly visible, or even visible at all. To relate this back to our example above, most information that could indicate phonetic variation is levelled by the normalisation. Nonetheless, some variation, above a certain threshold, is represented with special annotation. The VOICE transcription system defines a threshold up to which variation would be levelled. The threshold is defined along several factors. First of all we used a reference dictionary, the 7th edition of the *Oxford Advanced Learner's Dictionary* (OALD) (Hornby, Wehmeier, McIntosh, Turnbull & Ashby 2005) and secondly defined rules based on the count of syllables in a word (see Pitzl, Breiteneder & Klimpfinger 2008). Beyond the defined threshold of a changed number of syllables and thus regarded as representing significant variation and coinages, words are annotated as pronunciation variation and coinage (PVC) (cf. VOICE Project 2007). This method of normalisation cleans the information on the lexical level of variation, but sacrifices features relating to pronunciation, which were the primary focus in the design of the annotation system. Thus, for a user of the resource, it is very important to know and see how features are encoded in the corpus and be especially careful about possible claims. The assertion that there are only variations on the level of phonology which add or remove

at least one syllable would be obviously wrong, as this is one of the defined requirements triggering the annotation of the word as a PVC.

Possible methodology based syllogisms, as described above, have to be avoided as far as possible. They can in practice be far more subtle than the example of non-transcribed pronunciation features and some effort has to be put into avoiding these pitfalls. This can be remedied to some extent at the time of annotation as well as later when considering the methodology for analysis. Especially, when compiling and using an annotated corpus such as VOICE that is designed to be available to third parties, it is important that corpus compilers and corpus users take a shared responsibility and put considerable effort in understanding the implications of the available annotation. Thus, it is up to the compilers to provide annotation that is consistent, theory neutral and well documented, whereas corpus users have to be aware of and resolve possible conflicts with their own concepts, research questions and procedures of analysis.

## 2.4 Temporal Relation Between Source and Annotation

Any annotated research resource is meant to be used as a tool to investigate and – to a certain extent – understand the reality represented in this resource. Therefore, as noted above, annotation has to stand in a direct and reliable relation to the transcribed text or the recorded source. One aspect that is required for a meaningful interpretation of spoken language data is the temporal relation of the annotation to the spoken language. This means that the temporal relation of annotation to annotated source has to be unambiguous. Spoken language has an inherent temporal quality. This is mostly due to the fact that spoken language can be seen to be, at the lowest level, a temporally ordered sequence of sounds. Written language, in contrast, has only an implicit temporal relation that is expressed in the spatial representation of words and letters. It can, nevertheless, in most cases, be mapped to temporal aspects of spoken language.

The interpretation of the produced language, especially in non-monologic data, is tightly bound to temporal aspects. Thus, any meaningful interpretation depends vastly on the co-text established by the preceding, co-occurring and following language production of the participating individuals. Therefore, for any research that is interested in some aspect of the meaning of a particular language production – semantic as well as pragmatic – temporal aspects are indispensable for reliable interpretations.

Temporal aspects, here, refer to the order of occurrence. In contrast, for example, an analysis interested in the mere occurrence of tokens would be sufficiently served with a list-style representation of the tokens of the produced language in some format without any temporal information. For this purpose, it could be sufficient to provide an alphabetical list of identified tokens. From this list, it would be very difficult, if not impossible, to make valid claims on any level of meaning of a source text. That means that – to capture more of the meaning and get a more accurate view on the data – it is important to encode the

temporal range of transcription as well as the language tokens produced and a defined set of its features.

There are two general ways to establish a temporal relation in transcripts. First of all, they can be explicitly temporally aligned, that is the temporal relation is expressed by dedicated annotation. Secondly, the temporal order can be determined by the assumption that written language in itself resembles and establishes a relative sequential temporal relation that coincides intuitively with an auditive source . When reading transcribed texts, the assumption that the language generally appears in the presented order appears natural and unambiguous. Annotation that mingles with the text therefore also stands, by its spatial position, in a temporal relation to the text itself. In this abstraction, the transcript can be regarded as representing an annotation graph that is presented as a sequence of nodes which are connected by directed edges (cf. *figure 3* vs. *figure 5* and the discussion in *2.2. What is Annotation, Actually? p.7*). Nodes of these annotation graphs may be explicitly related to a given point in time of an input signal like an audio recording. Subsequent nodes that are connected via a directed arc in the annotation graph are therefore considered to be located at a point later or equal to the previous node on the time-line (cf. Bird & Liberman 2001: 48ff). Thus, any node $n_{i+1}$ that follows a given node $n_i$ is guaranteed to have a time $\tau_1 \leq \tau_{i+1}$. Bird and Liberman (2001) assume that annotation is non-hierarchical, which is – from a fundamental position, that is a position that is not determined by any particular data model – a sensible assumption. This implies that any annotation, as long as it is possible to be related to a time-line, can occur at any point of this time-line. This also includes the possibility that annotation can include or overlap other annotation. Yet, it can be useful to restrict the applicability of certain annotation at a given point. A basic restriction for an annotated spoken text would be for example that transcribed text cannot stand outside of an annotation that is marking a speaker contribution (i.e. `utterance`, see *4.2.12. utterance p.38*). This restriction imposes a hierarchy on the annotation by requiring a 'dominance' of one annotation over another. Even though this is a restriction and therefore a loss of flexibility, it is obviously useful as spoken texts outside of a kind of speaker contribution is unlikely to be meaningful. This kind of restriction is essential in two areas. First of all, hierarchical annotation like this can be validated at the time of creation. With the example of speaker contributions it would be a simple task to find transcribed text that is accidentally not 'correctly' put within a speaker contribution. Secondly, the temporal aspects can be useful and exploited by their structure in the later analysis. Nevertheless, it is important to see that this hierarchical restriction is just an additional requirement, but does not touch the possibility for establishing a temporal relation between annotations in this scenario.

We have seen that temporal relations are an important factor in the description of spoken language. They may be expressed, as described above, in different ways. That is, they may be expressed relatively to each other or with absolute references to a particular time-frame. In the following we explore some implications of the choice of either method.

### 2.4.1 Absolute Temporal Relations

Absolute temporal relations are relations that are established by reference to a common time-line. This time-line is shared by different parts of the annotation. When the temporal relation is established by defining absolute points on a time-line, every node in the annotation graph is associated with a time-stamp. A 'time-stamp' is defined as some measurement of time. That is to say that the temporal position of a node is determined solely by this time-stamp.

In practice, this means that the duration of some particular annotation, i.e. the length of the edge in the annotation graph, is established by the difference between its beginning and end. The advantage, and at the same time the main disadvantage, is that in this configuration every annotation has to refer to a measured point in time. This is an advantage since changes to any preceding annotation do not affect the position of the annotation. Furthermore,

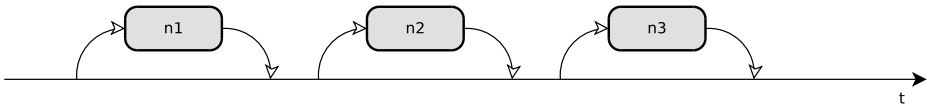



Figure 8: Annotation nodes are anchored at an absolute starting and end-point on the time-line. In this case, each part of the annotation is independent of each other and the sequence might only be re-established by comparing the values for the anchors of the nodes starting points.

explicit time-stamps allow for precise statements and inferences based on the relation of annotation to time. The major drawback, however, is that it is necessary to measure and enter time-stamps for all referenced points in time. Thus, this kind of annotation requires a lot more manual work for the measurement and alignment of annotation and time-line. This method should be considered if not only the relative order but the precise location of annotation in time are crucial for the planned analysis.

The temporal order has to be established by referencing a common time-line $t$ (see *figure 8*). The way this time-line is described is dependent on the annotated source material. An example for establishing such a common time-line for audio material would be to measure the time relative to the beginning of some recorded audio material. For transcribed or written texts as source material, it could be an offset measured in the number of characters from the beginning of the transcript. In both cases the order is established by relating to a common reference point. In practice, with spoken language, even though several implementations would be possible, annotation like this is usually maintained as a list with pairs of time and points in the annotation graph. The relation between the parts of the annotation can be established by comparing the time values of the points defining the annotation. That is, the points in time where some annotation begins and ends defines its duration and its sequential order. Possible overlap can be inferred from these absolute points in time (see *figure 9* and *figure 10*). Cases where the temporal overlap of elements

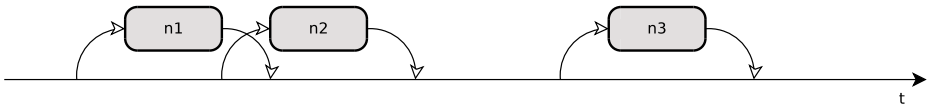



Figure 9: Annotation nodes are anchored at an absolute starting and end-point. In this figure, it becomes apparent that the temporal order of the annotation is only implied by the absolute time-line. Slight overlaps are generally unproblematic as the degree of overlap can be inferred from the values of the starting points.

is complete (see *figure 10*) and cases where elements overlap only partially (see *figure 9*) are technically the same when using absolute time markers. The only difference is in the relation between the starting points and end points of the overlapping annotation. Considering two annotations $A$ and $B$: The annotation overlaps completely when the starting point $t_{A1}$ of annotation $A$ is before the end $t_{B2}$ and start $t_{B1}$ of annotation $B$ while the end $t_{A2}$ is after the end of $B$. When only the starting point $t_{B1}$ or end-point $t_{B2}$ of the annotation $B$ is between $t_{A1}$ and $t_{A2}$ , the overlap is partial.

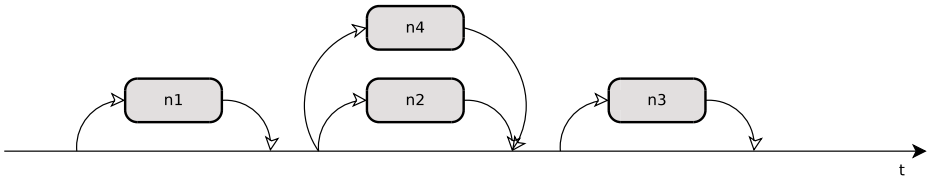



Figure 10: Annotation nodes are anchored at an absolute starting-point. Similar to slight overlaps (see *figure 9*), the order of the annotation is implied by the values of the staring points.

In this section we have seen that the temporal relation of annotation can be expressed by reference to an absolute time-line. In the following section, we will see that the same types of relation, that is sequence, overlap and partial overlap can be expressed with relative temporal relations.

### 2.4.2 Relative Temporal Relations

Temporal relations may not only be established by absolute markers on a common time-line, but also relatively to other annotation, i.e. as relative temporal relation (see *figure 11*). That is, the position in time is determined by some reference from one part of annotation to a preceding or following part of annotation.

This kind of temporal relation is very simple to establish in practice. Generally, it is possible to express a temporal sequence as a spatial sequence. This means that annotation of this kind is compatible with human reading behaviour, since it follows an analogous concept of ‘sequence’. Thus, relative temporal relations are not only simple to encode, but they are equally simple to decode for human readers. That is, it is possible to re-establish

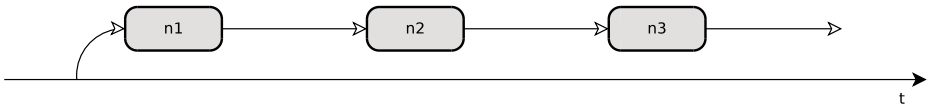



Figure 11: Annotation nodes are anchored at the beginning and then by reference to other annotation nodes. In this case, parts of annotation depend on other annotation to establish a sequence of annotations.

the order of annotations without having to rely on any particular tool, e.g. some specific software, but simply by reading the text.

The main potential disadvantage, however, is that absolute and precise measurements are not directly possible within this relative order. Even though precisely measured temporal information is not available, it is important to see that it is still possible to make valid claims based on the temporal embedding of the annotation. In fact, the temporal dependencies, such as the order of individual speaker's contributions, are more directly accessible as there is no need to compare references to a time-line. Whether this approach is suitable is, as with approaches favouring absolute temporal relations, a matter of the requirements for the research that should be possible with a resource. For uses where temporal order is needed to be established along the categories 'before', 'after' and 'simultaneous', relative temporal relations are sufficient.

Relative temporal relations allow for a similar granularity in the encoding of overlapping and partially overlapping events as absolute temporal relations. The difference between the two approaches is the chosen point of reference. Relative temporal relations are established by explicitly defining which portions of speech happen at the same time without precisely specifying at which point of time this is. Consequently, simultaneous events are straightforward to find, as it is not required to compare the times of all other annotations. When encoding slight overlaps, parts of the overlap have to be split in two parts to align to

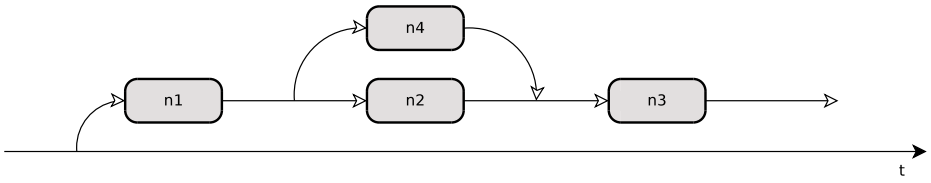



Figure 12: Annotation is established relative to each other. In this case a complete overlap of two annotations is established by having two parts of annotation referring to the same preceding and following nodes.

the segments that actually overlap. This renders the text-encoding slightly more complex in these cases as only complete parts of the annotation may be aligned. Nevertheless, these cases are relatively rare in VOICE. The complexity introduced at this point does not significantly reduce the main benefit of temporal annotation. That is, it is both easy to read

and good for querying. As relative temporal relations were more feasible in the preparation

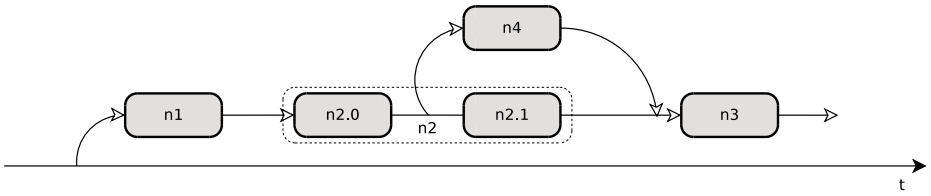



Figure 13: Annotation is established relative to each other. In this case a partial overlap needs to be encoded. This is a problem case, as the references are the same as in *figure 12*. That is, slight overlapping annotation is not possible without splitting part of the annotation according to the overlap (see node $n2$ ) and establishing a form analogous to *figure 12*.

of the transcripts as well as sufficiently expressive and more appropriate for the intended analysis, this approach is used for VOICE.

### 2.4.3 Possible Temporal Aspects of Annotation

We have already seen that temporal aspects are essential in the encoding of spoken language. The temporal dimension of annotations on the language level can be grouped into several different modes of temporal aspects. For this dichotomy, I introduced three descriptive categories. These categories divide annotation into three groups with similar temporal behaviour. The three categories I introduced and that are described in the following are

- eventive
- modifying
- temporally modifying

annotation. Every linguistic annotation in VOICE can be identified to be a member of one of these groups.

The first group is ‘eventive’ annotation. Eventive annotation for spoken discourse, as encoded in VOICE, can occur at any point where uttered words could occur. Nonetheless, this type of annotation is mutually exlusive to words. This means that a particular speaker can either produce some word-part or cause some event that requires eventive annotation, but not both at the same time. Prototypical annotation of this type are `pause` (see *4.2.23. pause p.40*) or `laughter` (see *4.2.17. laughter p.39*). In this respect, eventive annotation is similar to uttered words, regarding temporal aspects. Eventive annotation is defined by having a declared beginning, an end and thus a duration. These properties do not necessarily have to be explicitly quantified in any way. For an unambiguous interpretation, it is sufficient to define these properties as being an inherent by convention of some type of annotation. This way it is possible to establish a relative order of events within the transcript. Furthermore, eventive annotation and words are mutually exclusive, such that an

`utterance` (see *4.2.12. utterance p.38*) belonging to a given speaker cannot at the same temporal position feature lexical items and eventive annotation[5]. With the example of the `pause` this becomes obvious, as a given speaker can indeed either pause or talk.

The second group can be labelled 'modifying' annotation. Like eventive annotation, modifying annotation has a clear and explicit beginning. The duration and a determinable end are optional properties of this type of annotation. At this point, the similarities to eventive annotation is already exhausted as it is central for modifying annotation that it is not mutually exclusive with lexical items but can overlap their temporal position and include other annotation. Typical examples for modifying annotations in VOICE are shifts in voice quality as in the case of the speaking modes `soft` or `on_phone` (see *4.2.37. mode p.42*). As indicated above, the marking of an end of any modifying annotation is generally optional. Thus, this kind of annotation can be subdivided in two subgroups. The first group is annotation that does not only define the temporal position of the beginning, but also explicitly marks an end. The second groups omits a closing and relies on the reader or implementation to implicitly determine the temporal extent where the annotation is valid. In VOICE, most of the annotation in this second group requires an explicitly marked end[6]. Thus, it is much more convenient to validate the consistency of the produced mark-up if the end of the annotation has to be explicitly marked in the transcript. As a side-effect, explicit mark-up is reader friendly, as the reader does not have to keep the rules for the implicit termination of some particular annotation in mind.

The third type, 'temporally modifying' annotation is particular to annotation systems that do not strictly represent the temporal order in the physical order implied by the sequence of transcribed lexical items. I define this type of annotation as any annotation that does change the temporal order beyond the order implied by the typical reading flow. This definition is best illustrated with an example (see *figure 14*). In transcription systems that

**S4:** <fast> on a beer </fast> =
**S2:** = like <1> froth </1>
**S7:** <1> but </1> why is it <2> bad </2> when it (.) like (.) <ono> px </ono><whistles>
**S6:** <soft><2> yeah </2></soft>

Source: *Vienna-Oxford International Corpus of English* (VOICE 2009: LEcon562.1-4)

Figure 14: This example is taken from VOICE Online. It shows the use of temporally modyfing annotations (i.e. <1> and <2>).

adhere to rules like those defined in the VOICE *Mark-up Conventions* (VOICE Project

[5]It is important to note that in the annotation of spoken language, it is sometimes necessary to allow for temporal overlap between speakers. This facility is provided by the third group of annotation, 'temporally modifying' annotation.

[6]In the VOICE corpus format all modifying annotation requires an explicit end. In the VOICE transcripts there are very few exceptions, such as that speaker contributions (i.e. `utterance` see c_utterance) are implicitly ended by starting the next utterance with a `speaker_id`.

2007), the verbalised language is grouped into `utterance` units (see *4.2.12. utterance p.38*). These units strictly reflect the relative temporal order of the produced language. This is the default order a reader of a text-notation would expect for cases where the temporal relation of individual speaker's contributions is sequential. Therefore, a mechanism to highlight a deviation from this default is necessary for cases where the temporal order of the individual contributions can not be accurately represented. In this case, speakers' contributions overlap each other and thus break the sequence that is implied by the sequence of the contributions. For this case, 'temporal modifying' annotation supplies an appropriate representation as it allows deviation from a strictly sequential order. In VOICE, the annotation `overlap` (see *4.2.28. overlap p.41*) provides the means to annotate simultaneous speech and thus circumvents one of the general limitations of transcription in plain text-based formats. Annotation that would be inherently mutually-exclusive, like eventive annotation, can be annotated such that it is possible to represent different speakers that speak at the same time. Therefore, temporally modifying annotation is essential to the representation of spoken language.

## 2.5 Levels of Annotation

Up to this point, we have mainly discussed properties of the lowest level of annotation. That is, we have been concerned with annotation that stands in a direct relation to the produced language. Nevertheless, we will see that annotations can be divided into different layers and groups. Annotation can be used to express a great variety of different kinds of information. For example, some annotations are in place to encode the linguistic features (e.g. `pause` or `speaking_mode`). Other annotation is used to structure the text or encode information specific to a text-collection (e.g. `utterance` or meta-information within the text-header). To get to grips with the purpose of the vast number of different kinds of information encoded in annotation, annotation can, therefore, be divided into different layers and groups. A layer of annotation, here, refers to annotation with a similar scope. The term 'scope' refers to the textual range a particular annotation describes. The descriptive granularity of the annotation ranges from information describing the whole corpus resource (e.g. the list of individuals that appear as participants within the individual speech events) down to a level where the annotation describes the properties of a part of a word (e.g. emphasis). That means that any annotation that is descriptive of a speech event as a single entity can be regarded to be in one layer. Therefore, annotation that is descriptive of a particular scope, like annotation describing a stretch of language, can be regarded to be in a distinct layer. For a meaningful and valid interpretation of annotation, it is essential to be aware of the scope of the annotation. The general design of the annotation format of VOICE as expressed in the VOICE *Mark-up Conventions* (VOICE Project 2007)[7], as well as in the corpus format,

[7] A copy of the transcription conventions is included for reference in the appendix *VOICE Transcription Conventions (p.219)*

is generally designed to be explicit about the scope and the applicability of the annotation. Explicitness is a core quality, especially for linguistic annotations in VOICE. This type of annotation is, both in the VOICE Mark-up Conventions and the corpus format, realised as mark-up that is put directly within the text. As the mark-up that establishes the annotation is at the place where it is relevant, the relation to the text and the scope of the annotation is explicit.

The situation is somewhat different and complex for annotation that cannot be easily expressed directly within the transcribed text. An example for this kind of annotation is information on individual speakers like, for example, the age of a speaker. In this example, one has to know that the scope is the whole document and that the scope of the information is relevant to the document and all contributions of the particular speaker. Annotation with this type of relation to the text can be regarded as a layer that is half-detached from the actual language data. This is due to its position, as the mark-up defining the annotation is kept separate to the transcribed text, but is still essential to the interpretation of the source data.

The next level of annotation is information that does not at all have an anchor within the text, but is general information on the text. This is the layer of annotation that is not immediately present when reading through the annotated text. For example, this can be information on the resource, the encoding scheme, the project a resource was developed for and many more bits of information with a very broad scope. As this presents a level *above* the produced language, I refer to this kind of information by the widely used term 'meta-data'.

We have thus identified three distinct levels of annotation. First of all, annotation that is immediately present in the text. That means that it is defined by mark-up in the transcribed portions of speech. Secondly, annotation that is defined at a different place and thus has to be referenced. It can annotate one or more regions of the text. And thirdly, annotation that is a level above the language level and thus regarded to be meta-information.

I have chosen this bottom-up approach, as the annotation at text-level is most visible when dealing with the texts. Basically, text-level annotation is the layer of annotation that is immediately present when reading through the annotated text and usually the information that is most prominently featured and most important for the resource.

Any annotation that is represented as a segment of annotated elements can be transformed without information loss into a representation where the beginning and end of a segment are treated as an event-like entity. Nevertheless, only pairs of corresponding events can be rendered back into segments. Thus it is required for a loss-less and reversible transformation from segments to an event-like representation to include information on the start as well as on the end of the scope of any given segment-like annotation.

This chapter introduced concepts that are fundamental for the annotation of VOICE. In the following chapters the VOICE transcription-system will be described as a format that

is built on previous mark-up schemes and a formal definition of the mark-up syntax that is used for manually transcription and annotation.

# 3 The Legacy Format from the Pre-VOICE Era

For VOICE, the description of its predecessors is more important than for many other corpus projects. VOICE deviates from the trodden paths in that the transcription format is not identical with the targeted corpus format. Instead, it uses an intermediate format for transcription which was later transformed to the corpus format. Most other corpora are assembled directly from transcripts in the final format. This might be either plain text, as it is often used in older corpora, like the Brown Corpus (cf. Francis & Kucera 1979) or a more complex format like an XML based schema. Latter strategy can become very difficult to handle, as complex mark-up that is not specifically designed for transcription, but with an eye on the later evaluation of the resource, is usually difficult to read and write. There are basically two possible approaches in handling a greater degree of complexity in manual transcription. It is either possible to use a specific editing sofware that hides the transcription format, or to define a mark-up language where many of the relations are implicitly defined by its specific syntax. The first hides the data behind a simplified representation of the file format through a graphic user interface. The latter defines a specific mark-up language for transcription. This option also has the advantage that it is only required to set up the basic syntax of the transcription mark-up language to get started with transcription, without any direct requirement to build a special software for editing the transcripts. The mark-up language for VOICE is based on its predecessors, albeit much refined and specifically designed to be machine-readable and transformable to a specialized corpus format. Thus, it is important to describe some basic aspects of the transcripts for projects before VOICE and the way main concepts are transferred to the final transcription format.

## 3.1 Description

As most large-scale research projects, the VOICE Project was preceded by several smaller data-oriented studies in its field. These projects, apart from serving the purpose of particular research-questions, were de-facto piloting the later VOICE Project. These projects showed that the systematic collection and transcription of ELF data is feasible, worthwhile, promising new insights and is also possible in a larger scale project like VOICE. The mark-up devised for these projects provided a good proof of concept and, furthermore, served the analytic purposes of several individual papers and diploma theses[8]. Nevertheless, the requirements for these pilot studies were different from the broader corpus-perspective

[8]See for example: Kathrin Kordon 2003, Marie-Luise Pitzl 2004, Thomas Strasser 2004, Angelika Breiteneder 2005 and Theresa Klimpfinger 2005

of the VOICE project. Therefore, the transcription format was built for the individual, mainly qualitative interpretation of data, where possible restrictions in scalability towards larger corpora was not an issue. Nevertheless, the predecessors of VOICE proved to be an invaluable source of expertise and transcription experience. In this respect they have been valuable especially for the definition of the VOICE transcription system.

While most of the mark-up was self-explanatory for a scientist working with transcripts in the field of linguistics, the transcription format of individual projects were neither fully documented nor formally defined. This was sufficient (and certainly appropriate for the analysis of the data) for the scope and intent of the individual projects. Nevertheless, without a formal definition of the mark-up it could neither be automatically tested for consistency or plausibility nor could the encoded information be extracted automatically. For the pilot projects, this was not a requirement, as the amount of data was limited and the individual researchers had enough general knowledge of the creation history of the resource and thorough experience with the individual texts to find erroneous tagging. But for a large scale project it is a requirement to have a defined and unified transcription format. The amount of data in a large scale project and the complexity of mark-up requires defined mark-up restrictions to be in place. That is to say that some specific annotation can only occur at well defined points in the transcript (e.g. a word has to be part of a `utterance`. This is important for larger projects, as it is far more difficult to find mark-up errors and inconsistencies in a huge resource. In the pilot projects, the data was mostly used to exemplify and show particular linguistic phenomena in context. This use as supportive evidence allows for less rigid mark-up as the interpreter, the human researcher performing mostly qualitative research, is capable of understanding the mark-up even if it is not entirely consistent. Computer aided processing would obviously fail in many cases to disambiguate inconsistent mark-up and consequently provide the researcher with possibly wrong or at least ambiguous figures. Nevertheless, a use of this kind was not intended for the pilot studies, but the experience was invaluable for refining and redesigning the transcription conventions for a project with a much bigger scope.

## 3.2 The Transcripts

The transcripts from pilot projects where stored in Microsoft Word format. The Word files were used mostly as a container for generic plain text annotation which was in some cases made more readable by enriching the annotation with some formatting like different font faces and colouring. Most of the mark-up was represented textually, that is, represented by distinct sequences of characters. This formed the body of the transcribed text supplemented with basic information on circumstances of the given event.

### 3.2.1 Description of the Files

Most of the transcripts of the pre-VOICE era were keyboarded into an ordinary word-processor producing Microsoft Word Documents (.doc)[9]. Word Documents are a proprietary file format designed to store information for document preparation for desktop publishing. Thus it is most suitable for the preparation of data for printing. Therefore, the file-format can be a good choice if a researcher wants to prepare data sets and format small parts for inclusion in individual research papers. If the task is to consistently annotate bigger sets of data for a more general audience, the restrictions of the format become evident. At the time of keying, writing the transcripts with an ordinary word processor might be appealing, as most researchers feel comfortable with the tools they use for various other tasks in their everyday life. But, as soon as the data needs to be further processed, the file-format becomes an issue. The main problems with word-processor documents are:

- Text is difficult to extract.
- Annotation is difficult to access.
- Mark-up that does not have a textual representation, but a style (i.e. a particular font face, style or colour), is almost impossible to extract with common tools designed for linguistic analysis.
- The encoding of the file's content may allow for characters not desired in or differently encoded in the final corpus.
- Word processor documents are not usable with most software available for linguistic research.
- While it may be desirable to have written mark-up also displayed in a specific style (like VOICE currently uses blue for overlapping speech), using a word processor, this requires the transcriber also to manually add style information to a portion of a given transcript. This doubles the work for the transcriber for a given piece of mark-up and adds the danger of inconsistencies. See *4. Redesigned Transcription Format (p.29)* for a discussion on how the VOICE transcription format deals with these issues.

The choice of Microsoft Word files as a container for transcripts of spoken language for linguistic analysis is not optimal. Nevertheless, much of the structure, concepts, the textual representation of annotation and the specific mark-up could be retained. Especially the notion of the document as the main unit representing a transcribed speech events is useful and continued in VOICE.

### 3.2.2 Data Arrangement

The data of the textual representation of pre-VOICE transcripts usually consists of three parts:

[9] The transcript files were usually stored in the Microsoft Office 97/2000/XP format

- A header with information on the transcriber, dates, duration, extent.
- A transcript body, containing the written annotated representation of the spoken language.
- A footer with comments on the transcription process and possible difficulties. This part, however, only features in few pre-VOICE transcripts.

The three-part basic structure of the manuscripts is only loosely maintained between different pilot projects and a great degree of variation is encountered. The later pilot transcripts have a strong tendency to this general content organisation, while earlier transcripts greatly differ in their transcription practice. This is an indicator of the fact that the transcription developed and matured on what appeared to be useful and feasible to the transcribers, who at the time were identical with the researchers preparing the data to answer particular research questions. In the following sections, the most important mark-up features are described. Note that a great degree of variation is present between different encoders of transcripts.

### 3.2.3 The Header

The header of pre-VOICE transcripts is usually composed of several lines containing each a pair of a label and a value. These lines generally provide information on the length of the event, the duration of the transcribed portion of the text, the initials of the transcriber, the transcription date and a title.

```
Project:  VOICE
Title:  UNICADEC.part2
Date:  9.12.2004
Duration:  70 min.
Duration of transcription:
Transcription by:  AW
Transcription date:  March 2005
Number of words transcribed:
```

Figure 15: Header from a late pre-VOICE transcript

A transcript header does nevertheless not encompass all supplementary information available for a given event. Most events are accompanied by field notes, lists of speakers and descriptions relating to the setting of the event. This supplementary information is stored in separate files next to the main transcript file. Information stored alongside the main transcript file is therefore not immediately present and has to be manually retrieved when necessary for individual research tasks. For small pilot studies, this is not an issue, but had to be redesigned for the published VOICE corpus. That is, for VOICE it is a requirement to have the data immediately accessible for analysis. Nevertheless, information that is solely

required by the transcriber for the purpose of transcription and not for publication is still to be found in separate files.

### 3.2.4 The Transcript Body

The body of the transcripts is as little standardised as the header and the main structure of the transcripts. Nevertheless, one common feature of all pre-VOICE transcripts is that a body section is present which is optionally surrounded by meta-information and notes. Usually the notes follow the body and the meta-information on the event precedes the body in a header section.

The body holds the main part of the information, namely the transliteration of the spoken language, including mark-up for various features of the captured language. The transcribed features vary greatly among transcripts, depending on the particular research requirements of a particular pre-VOICE project. Here, as with the basic layout of the transcripts, the accumulation of know-how from project to project can be noticed, which leads to the later transcripts becoming more comparable to each other in regard of structure and mark-up.

The basic format of the text body is very similar among the different pre-VOICE projects. None of the pilot projects used a partitur based notation[10], but all employ an intuitively readable stage-like notation. In pre-VOICE transcripts, each utterance[11] line is preceded by a sigle identifying the speaker by a label. This makes it possible to reference different utterances to a single speaker and identify the speaker in the accompanying material. The remainder of the line is the transliterated language produced by the speaker, enriched with special mark-up. This mark-up is fairly consistent within any given pilot project but not necessarily identical across individual pilot projects of VOICE. The general concepts encoded in the pre-VOICE pilot projects are very similar. Some aspects, however, differ greatly in the chosen mark-up vocabularies.

### 3.2.5 Transcriber Notes

One feature that was generally required in pre-VOICE pilot projects is the use of transcriber notes. Transcriber notes are notes collected during the process of transcription. The notes range from short comments to record open questions on the transcript to analytical notes for the later scientific use. Notes of this kind appeared to be and later proved to be a very valuable resource. This way, information on specific difficulties a specific event poses to the transcriber can be easily passed from one stage of the transcription to the next. Many of the pre-VOICE texts were created in small-scale projects where usually only one person was involved in the creation and proofing of the texts. For these projects the transcriber-notes were in fact not as important as for a large-scale project like VOICE. Therefore transcriber

[10] 'Partitur' based notation features a structure similar to a musical score. The term became popular with the Exmaralda transcription tool (see Schmidt 2001: 223)

[11] 'Utterance' in this context refers to a stretch of speech by a speaker up to the point where the speaker pauses and another speaker starts (see *Mark-up Conventions (p.219)*).

notes became a mandatory part of VOICE transcripts, as they serve the communication between different annotators.

# 4 Redesigned Transcription Format

The pre-VOICE transcripts are transcribed in formats with many similarities. Nevertheless, from a formal perspective, these transcripts follow slightly different encoding practices. One main tasks that determined the early stages of the VOICE Project was, therefore, to reorganise and unify the transcription formats. This process started even before the main phase of the corpus compilation. With this prerequisite in place, consistent transcriptions that meet the requirements for VOICE, could be produced.

The VOICE team decided very early to keep a transcription format similar to the transcription format of many of the pilot projects. The main implication of this decision is, that the VOICE transcription format follows a text based notation. Consequently, mark-up for the definition of the annotation and the spoken words are transcribed simultaneously within the same text file. This decision also implies that the transcripts are produced in a format specific to VOICE. A more generic approach would be to have the transcriptions in an XML format similar or even equal to the format of the targeted corpus format. Even though VOICE uses an TEI-XML based file format for file-storage, data investigation and retrieval for the corpus, the transcripts are not in XML. Nevertheless, many elements of the transcript follow a pseudo-XML notation. 'Pseudo-XML' means that the mark-up follows a structure similar to XML. The pointy brackets around tag names (e.g. `<loud>`) are a good example for structural similarities. Nevertheless, not all rules for well-formed XML are met. For example an element `<1>`, indicating overlapping speech, would not meet the requirements for XML. The tag name (i.e. "1" in this case) cannot start with a numeric character. Furthermore, the general document structure is different and, most importantly, not all annotation is encoded in pseudo-XML. The transcript adheres to a carefully defined plain text format that uses few special characters to separate mark-up and text. Due to the formalisation of the VOICE transcripts, they feature a structure that can be mapped to a more explicitly structured XML representation with moderate effort. Mainly, this means that the transcripts are in a different format than the resulting corpus. This is due to the requirement that a corpus should be stored in a format that can be readily processed by advanced tools or tool-chains and is not tailored to be used exclusively with one specific tool. Transcripts, in contrast, should follow a format that makes the work of human transcribers as easy as possible.

One of the main benefits of using a plain-text based transcription format is that transcribers who are already familiar with the VOICE predecessors or other text based transcription systems can start transcribing immediately. Drawing on their previous experience, they can transcribe with very high accuracy, compared to the relatively low

training effort. Secondly, VOICE features highly interactive speech events, with a rich set of annotations. Encoding all mark-up manually in XML could easily obscure the transcriber's view on the text. XML would lead to relatively verbose mark-up which leads to an increased perceived complexity. Thus, this would make the process of transcribing and proofing the texts more difficult and time-consuming. Nevertheless, XML based transcription would allow for simple and straightforward validation on well-formedness as well as against a specific schema. Furthermore it would be possible to base the tools for processing the files with existing implementations and standards. The drawbacks of having to find specific solutions for a text based, rather than an XML based transcription, have to be acknowledged. The points that have to be addressed are the formal validation of the transcripts and the development of a tool-chain that allows for the processing of input documents. Nevertheless, for the VOICE project, the benefits of maintaining a plain-text based transcription appeared to outweigh the overhead of creating specific non XML based tools for the transcription and the later transformation of the transcripts to the XML based corpus format.

The following criteria were used for the development of the structure of VOICE transcripts. They should be:

- based on previous conventions
- easily readable
- writable by trained transcribers with moderate training
- unambiguous for interpretation
- capable of expressing the whole range of annotation
- machine readable

One of the main objectives of the VOICE *Mark-up Conventions* (VOICE Project 2007) is to achieve a high reliability of the produced transcripts. As the annotation is directly written into the text, it is important to keep the transcripts, nevertheless, readable. Good readability, here, includes the aspect that the encoded flow of conversation should be accessible and easy to follow. This aspect is not only important at the stage of actual lexical transcription and annotation, but also in the subsequent proofing phases. Yet, this implies that the mark-up should, even though it adds much information to the plain text of the transcription, not disrupt the flow of reading too severely. A transcription format featuring good readability and thus one which honours the aim to keep the flow of conversation visible, enhances the understanding of the transcriber and makes the transcription process more reliable.

Maintaining the ability to fluidly read the texts can be established by keeping the mark-up very short and obviously different in shape to lexical items. Edwards (1992) identifies this in her summary on principles for the visual display of transcriptions of spoken language in *Design principles in the transcription of spoken discourse*. The three related concepts are "proximity of related events" (Edwards 1992: 131), "efficiency and compactness" (Edwards

1992: 131) and "visual separability of unlike events" (Edwards 1992: 131). With very short mark-up, the distance between the lexical items becomes short enough to be bridged or skipped over while reading. Nonetheless, very short mark-up can also be problematic, as numerous short bits are more difficult to remember. A certain degree of redundancy is in practice not just "non-essential and distracting clutter in the transcript" (Edwards 1992: 33). Therefore,

> [Edwards'] principle concerning efficiency and compactness [...] should be applied with some caution, as it often seems favorable to maintain a certain amount of redundancy. (Bruce 1992: 145)

VOICE features over seventy distinct types of annotation that are in many cases composed of several mark-up tokens. It is a prerequisite to remember the individual mark-up items for using them in transcription. Therefore, great care had to be put on the mnemonics of the mark-up. That means that the mark-up is designed with a focus on being easy to remember for reading and writing VOICE transcripts. With this mark-up, frequent annotations are encoded as well as very rare features. The frequency distribution ranges from the almost omnipresent pauses[12] to relatively rare features like the indication that a speaker loudly reads some text[13]. The difference in the observed frequency between the most and least frequent mark-up types are about three orders in magnitude. Therefore, it is apparent that VOICE mark-up and the strict application of the VOICE Transcription Conventions lead to a wide frequency distribution of individual mark-up tokens. Frequently occurring mark-up tokens, like pauses, are part of the VOICE mark-up as well as far less frequent, albeit equally important tokens like speaking modes. This wide distribution in frequency leads to very different requirements concerning the brevity of the mark-up. While very concise mark-up for very frequent annotations yields a great reduction in the size of mark-up, the effect of keeping infrequent mark-up small is almost negligible in the overall picture.

The readability and writeability is first of all ensured by keeping frequent mark-up as short as possible. Short mark-up is beneficial as it does not obfuscate the text beyond the necessary. Thus, a flow in reading and writing can be maintained and the danger of losing the overall orientation within the text is significantly reduced. This focus on readability is important both for transcription and later analysis. The frequent brief pause, for example, is encoded as "(.)". Thus, a brief pause requires three character entries with a normal text editor. Furthermore, the work-load that frequently used mark-up imposes on the transcriber can be further reduced by implementing specific keyboard macros. Therefore, VoiceScribe (see *VoiceScribe p.106*) features short-cuts for the most frequently used annotations in VOICE. Even though keyboard macros improve the writing speed, they have no influence on the readability. On contrary, as they allow to insert long mark-up with only few key-presses, they may tempt the designer of a mark-up system to increase the mark-up size. To

[12] VOICE contains 112276 pauses.

[13] VOICE contains 250 shifts to the speaking mode `reading_aloud`.

keep the transcripts readable, it is important to still keep the frequent mark-up items small, even though one can not save as much with short-cuts as with long mark-up. The long and infrequent mark-up items on the other hand do not justify the definition of a special keyboard macro as shortcuts that are very infrequent would not likely be remembered and used. Therefore, looking back to above example, the brief pause, is in VoiceScribe entered by pressing two subsequent full stops. With only two key-strokes, the mark-up is entered immediately. The situation is similar for other frequent mark-up, such as overlaps.

Not only the brevity of frequent mark-up keeps the texts manageable, but also the explicit longer notation of less frequently used mark-up. The speaking mode "reading aloud", being relatively rare compared to other mark-up, is therefore encoded "<reading aloud> ... </reading aloud>". The less frequent mark-up is self explanatory, which renders the mark-up easy to remember for a transcriber writing the transcript and also comprehensible to a (proof-)reader of the text.

Both the brevity, on the one hand, and the explicitness, on the other, shorten the training period required to master the rich set of annotation. Thus, more time can be spent on accurate training of the interpretative and linguistic aspects of transcribing audio recordings of highly interactive speech events.

Furthermore, much of the cross-referencing that is present in the resulting XML corpus format can be established automatically, such that the transcriber can focus on annotation that requires linguistic interpretation and understanding of the circumstances of an event. Basically, the transcription process is designed to keep repetitive work that may be easily done by a machine away from the transcriber. The transcriber can therefore spend more time and thought on the actual transcription.

## 4.1 Formalisation of VOICE Transcripts

VOICE transcripts are the main data source for the corpus. They represent the produced language, the temporal relation between produced speech and the linguistic features identified and represented during transcription. A secondary data source, a database, is used to hold information on events, individuals and speakers. This is the source for merging additional information into the transcribed speech when it is transformed into the XML corpus format (see *6.5.4. Inclusion of Related Data From Non-transcript Based Data Sources p.93*). Thus, meta-information and transcribed speech is then available to the researcher from the individual corpus files.

The structure of the transcripts follows a tree-like paradigm. This means that any component of a transcript may either be composed of other components or be a terminal component. Terminal components differ from other components only in so far as they are not composed of components themselves. The grammar of the mark-up for VOICE is designed to be recursively parseable. This means that a parser starts with the initial assumption that it reads a document which follows the mark-up conventions for VOICE.

Therefore, the text has to be composed of the parts that are required for a `transcript`. The parser analyses the components that form any given component in VOICE by the generative rules and the components required to fulfil this rule. This process is performed

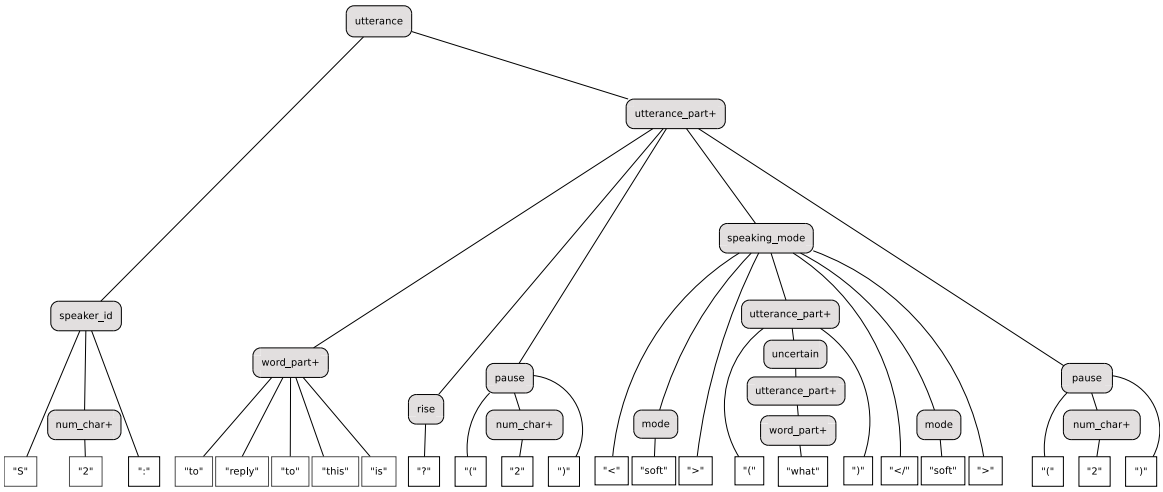



Figure 16: Parse tree for an `utterance`

to a level where there remains no component that is itself composed of other components. The result of this would then be a tree-like structure, the parse tree (compare *figure 16*). The root of this tree – that is the initial point or starting point of the parsing – is the `transcript`. Branches and leaves are defined by the corresponding component of the generative model. There, terminal components are represented as the leaves and the nesting of non-terminal components define the joints and branches. See *4.2. Components of VOICE Transcripts (p.33)* for the definition of the transcript format and a brief description of individual components.

## 4.2 Components of VOICE Transcripts

The transcripts are, like many of the transcripts from pilot projects, divided in three main components: The header, the body and the transcriber notes. This indicates the most important aspect of VOICE transcripts. They are assembled from components that relate to each other. The relation is, as in natural language, paradigmatic and syntagmatic. That is, the relation follows a defined grammar. In this section the inner structure of the transcript is formally defined in a format based on the Backus-Naur Form (BNF) type of notation. The term 'Backus-Naur Form' may require some explanation. It refers to a specific notation for the definition of a formal grammar. As Knuth (1964) pointed out, similar concepts have been formalised by other scientists. He notes that an

> [...] equivalent type of syntax has been used under various other names (Chomsky type 2 grammar, simple phase structure grammar, context free grammar, etc.). There is still a reason for distinguishing between these, however, since linguists present the syntax in the form of productions while the Backus version has a quite different form. (Knuth 1964: 735)

This quote points to the main benefit and probably the reason why BNF is in variants used by several standardisation bodies to define computer languages, namely that it has a concise and readable and comprehensible syntax. The variant of the BNF used for the specification of the transcription language for VOICE transcripts is based on the extended BNF (EBNF) notation as used by the World Wide Web Consortium (W3C) (cf. Bray, Paoli, Maler & Yergeau 2008).

The following definitions are of main interest for researchers building their own tools upon the *Mark-up Conventions* (VOICE Project 2007). Thus they are meant to be read alongside these conventions, as the practicalities and transcription rules employed in the process of transcription are described there. It is important to know that additional consistency checks are enforced by the current VOICE Parser[14] that are based on the 'soft rules', meaning the encoding practice described in *Mark-up Conventions* (VOICE Project 2007) in contrast to the encoding possibilities defined in below table.

| Component | | Definition |
|---|---|---|
| transcript | ::= | transcript_start, header, body, transcriber_notes |
| transcript_start | ::= | "VOICE" |
| header | ::= | short_title, date_of_evt, header_component* |
| short_title | ::= | "Short title: ", alpha_num+ |
| date_of_evt | ::= | "Date of event: ", date |
| header_component | ::= | header_label, ": ", header_spec[15] |
| header_label | ::= | alpha_num+ |
| header_spec | ::= | date \| alpha_num+ |
| body | ::= | (medium \| gap \| nrec \| noteref)+ |
| medium | ::= | medium_start, (track \| noteref \| utterance)*, medium_end |
| track | ::= | "<track ", medium_disc, "_", medium_track, "_", medium_time, ">" |
| utterance | ::= | speaker_id, utterance_part+ |
| utterance_part | ::= | space[16] \| word_part \| pause \| lengthening \| uncertain \| unintelligible \| contextual \| rise \| fall \| foreign \| laughingly \| noteref \| directed \| track \| overlap \| other_cont \| onomatopoeic \| speaking_mode \| speaker_noise |
| speaker_id | ::= | "S", (num_char+[17] \| ("X", ("m" \| "f" \| ("-", num_char+[18]))? \| "S" )), ": " |

[14] See *VOICE Parser (p.106)* for further information on the implementation of the VOICE Parser.

[15] Sample components include, "Duration of whole recording", "Transcript duration", "Number of words transcribed", "Number of Speakers", but might also include more components to capture additional information on the text or fine-tune the implementation of a transformation to XML.

[16] Space is handled differently in utterances as it delimits individual words.

[17] In VOICE a maximum of two digits is allowed for the speaker_id.

[18] This number has to reference a speaker ID that is present in the transcript.

| | | |
|---|---|---|
| space | ::= | (" " \| "\t")+ |
| word_part | ::= | (alpha_uchar \| alpha_lchar \| apostrophe \| laughter \| hyphen \| anon \| spelt \| pvc \| year)+ |
| laughter | ::= | "@"+ |
| hyphen | ::= | "-" |
| anon | ::= | "[", alpha_num+, "]" |
| spelt | ::= | "<spel>", (space \| alpha_char \| rise \| fall \| anon \| noteref \| lengthening \| track \| uncertain \| pause \| overlap \| laughingly)+, "</spel>" |
| pvc | ::= | "<pvc>", (ipa \| pause \| uncertain \| overlap \| alpha_char \| hyphen \| rise \| fall \| space \| lengthening \| noteref \| laughingly \| speaking_mode \| track)+, desc_trail?, "</pvc>" |
| year | ::= | num_char, num_char, num_char, num_char |
| pause | ::= | timed |
| timed | ::= | "(", ("." \| num_char+), ")" |
| lengthening | ::= | ":" |
| uncertain | ::= | "(", utterance_part+, ")" |
| unintelligible | ::= | "(", unint_syl+, ")" |
| contextual | ::= | desc_trail |
| unint_syl | ::= | (word_char_x \| word_char_X \| rise \| fall \| lengthening \| hyphen \| ipa \| speaking_mode \| pause \| noteref \| overlap[19] \| laughingly) |
| overlap | ::= | "<", num_char+, ">", utterance_part+, "</", num_char+[20], ">" |
| word_char_x | ::= | "x"+ |
| word_char_X | ::= | "X"+ |
| other_cont | ::= | "=" |
| onomatopoeic | ::= | "<ono>", (ipa_char \| space)+, "</ono>" |
| rise | ::= | "?" |
| fall | ::= | "." |
| foreign | ::= | "<L", ("1"\|"N"\|"Q"), (alpha_char, alpha_char, alpha_char?), ">", (word_part \| space)+, "</L", ("1"\|"N"\|"Q"), (alpha_char, alpha_char, alpha_char?), ">" |
| | ::= | "<ipa>", ipa_char+, "</ipa>" |
| speaking_mode | ::= | "<", mode, ">", utterance_part+, "</", mode, ">" |

[19] It has to be assured that only characters valid in `c_unint_syl` are used (i.e. “x” and “X”), as the content model of overlap allows for a broader set of characters.

[20] The second `num_char` has to correspond to `num_char` at the beginning of the overlap.

| | | |
|---|---|---|
| mode | ::= | "fast" \| "soft" \| "slow" \| "loud" \| "whispering" \| "sighing" \| "singing" \| "yawning" \| "reading" \| "reading aloud" \| "on phone" \| "imitating" \| alpha_char+[21] |
| laughingly | ::= | "<@>", utterance_part+, "</@>" |
| speaker_noise | ::= | "<", noise,(" ", timed)?, ">" |
| noise | ::= | "sighs" \| "squeals" \| "applauds" \| "coughs" \| "clears throat" \| "sneezes" \| "snorts" \| "yawns" \| alpha_char+ |
| directed | ::= | "<to ", speaker_id, ">", utterance_part+, "</to ", speaker_id, ">" |
| medium_start | ::= | "<beg ", medium_disc, "_", medium_track, "_", medium_time, ">" |
| medium_end | ::= | "<end ", medium_disc, "_", medium_track, "_", medium_time, ">" |
| medium_disc | ::= | alpha_char, alpha_char, alpha_char?, num_char, num_char?, alpha_char? |
| medium_track | ::= | num_char, num_char? |
| medium_time | ::= | short_time |
| gap | ::= | "(gap ", long_time, ")", desc_trail |
| nrec | ::= | "(nrec ", long_time, ")", desc_trail |
| long_time | ::= | num_char, num_char, num_char?, ":", num_char, num_char, ":", num_char, num_char |
| short_time | ::= | num_char, num_char, ":", num_char, num_char |
| desc_trail | ::= | "{", (alpha_num[22] \| space)+, "}" |
| noteref | ::= | hash \| excl |
| hash | ::= | "<#", num_char+, ">" |
| excl | ::= | "<!", num_char+, ">" |
| transcriber_notes | ::= | "<transcriber_notes>", general_note?, (hash_note \| excl_note)*, "</transcriber_notes>" |
| general_note | ::= | (alpha_num \| space)+[23] |
| hash_note | ::= | "<#", num_char+, ">", general_note, "</#", num_char+, ">" |
| excl_note | ::= | "<!", num_char+, ">", general_note, "</!", num_char+, ">" |
| alpha_num | ::= | alpha_char \| num_char |

[21] Speaking modes are parsed relatively late, therefore it is possible to have an open list of possible speaking modes. Nevertheless, it is recommended for an application to emit warnings whenever other speaking modes are used in a transcript. Obviously, it is not allowed to use modes that would lead to the speaking mode resembling other mark-up. Examples are "ipa" or "ono", which are reserved for the respective tags.

[22] The current implementation of VOICE does not restrict to `alpha_num` but allows any UNICODE (cf. Unicode Consortium 2010) character except for a literal "}".

[23] In practice, much more than alpha-numeric characters are allowed here. In transformation, care has to be taken not to use characters that might indicate either the end of the note or of the `transcriber_notes` component.

| | | |
|---|---|---|
| alpha_char | ::= | alpha_lchar \| alpha_uchar |
| alpha_lchar | ::= | "a".."z" |
| alpha_uchar | ::= | "A".."Z" |
| apostrophe | ::= | "'" |
| date | ::= | num_char, num_char, num_char, num_char, num_char, num_char, num_char, num_char |
| num_char | ::= | "0" \| "1" \| "2" \| "3" \| "4" \| "5" \| "6" \| "7" \| "8" \| "9" |

### 4.2.1 transcript

The transcript component represents the complete transcript of a given speech event. The content model of the transcript component includes `transcript_start`, `header`, `body` and `transcriber_notes`.

### 4.2.2 transcript_start

Any VOICE transcript starts with the literal string "VOICE". This can be used to distinguish VOICE transcripts from other text-files, but does not carry any other information apart from this.

### 4.2.3 header

The first part of the transcript is the header. It is the right place to capture information on the state of the file, a revision history with custom `header_component` components. Mandatory elements are the `short_title` and the `date_of_evt` described in the following two numbered paragraphs. Nevertheless, it is useful to make other information like the number of speakers or responsibilities in the creation and proofing of the transcripts, mandatory in a given implementation. See the VOICE transcription conventions for more details on headers required in VOICE.

### 4.2.4 short_title

The short title is a string that identifies a particular event. In VOICE this information has to be unique for any given transcript and is used to relate and merge information kept in the database with the transcripts in the conversion to VOICE XML.

### 4.2.5 date_of_evt

The date when the event was recorded.

### 4.2.6 header_component

Any other information that should be given in the header. This information consists of a `header_label` for identification and the information as such. Most common are `header_components` relating to the transcription and proofing process or information on speakers and the languages they use.

### 4.2.7 header_label

An alpha-numeric string identifying some information given in the header. This is a very open and broad definition of the label. In implementations, such as the VOICE transcript parser, this should be maintained in a closed list. This way it is possible to restrict the labels to values that can be reliably processed.

### 4.2.8 header_spec

Either a date or an alpha-numeric string. The type 'date' is in fact a subset of an alpha-numeric string. It is still explicitly mentioned here, because of its frequent use in `header_components` referring to the transcription and proofing process and is treated separately by the VOICE transcript parser.

### 4.2.9 body

The main part of the transcript. This part consists of the transcription based on audio recordings of the speech events. The body consists of blocks of transcription that correspond to a specific input medium (i.e. a MiniDisc™) and possible gaps in transcription.

### 4.2.10 medium

Mediums are the main building blocks of the transcription in the transcript body. They reference a media file and supply temporal information on the beginning and end of the transcript-block. Note that all utterances in VOICE transcripts directly belong to a medium component.

### 4.2.11 track

Indicates the transition to the next track on the medium. This annotation loosely aligns transcript and audio source. The information in track components can be used at a later stage when the individual components of the transcript or the derived VOICE XML document could be temporally aligned.

### 4.2.12 utterance

Any portion of speech that is produced by a single speaker or by a group of speakers that is referenced with a sigle. The utterance contains the lexical transcription and additional components to annotate pauses, lengthening, uncertainty, unintelligible speech, contextual information, extraordinary rise and fall in intonation, switches into other languages than English, speech in speaking modes like laughingly speaking or speech on phone, references to transcriber notes, directed speech, overlaps, other continuation, onomatopoeic language and noises the speaker produces.

### 4.2.13 utterance_part

This is the main content model organising the transcribed speech. See the description of utterance for details on the allowed content.

### 4.2.14 speaker_id

The `speaker_id` is a sigle that prepends every utterance and is used in anonymisations to replace the name of a speaker. It is unique within a given event and identifies a single speaker. Exceptions from this rule are the special `speaker_id` "SS" for utterances that are assigned to the group of interactants and "SX" for utterances that were not possible to be assigned to a speaker. In uncertain cases also the variants "SX-f" or "SX-m" are used to at least provide information on the gender of the speaker. "SX-1" would be used if the speaker is probably "S1", but could not identified with sufficient confidence by the transcriber.

### 4.2.15 space

In VOICE, transcribed words are delimited by space. This means that any spacing character that occurs within an utterance, indicates the potential beginning of a new word. The first `word_part` after a space establishes the beginning of a new word.

### 4.2.16 word_part

This component defines the elements that can form words. The most important types that can represent this component are alphabetical characters, hyphens, but also components which constitute themselves by definition a word. These are for example anonymised or spelt words, pronunciation variation and coinages and laughter.

### 4.2.17 laughter

Laughter is encoded by the AT-symbol "@". The number of AT-symbols in sequence is determined by an approximation to an equivalent syllable-like unit and is to be read as an estimate of the duration.

### 4.2.18 hyphen

The hyphen is used where it is required by the spelling conventions. Furthermore, the hyphen is also used to indicate incomplete words. This feature is very frequent in spoken language and is indicated by a trailing hyphen appended to the `word_part` Not only words that are shortened by interruption or self-interruption may be indicated by a hyphen, but also the continuation of a previously interrupted word by a leading hyphen.

### 4.2.19 anon

Anonymised words are replaced by a supplied label in square brackets. Common labels in VOICE are speakers' first and second name with their respective speaker ID and abbreviation for organisations like "org1" are possible.

### 4.2.20 spelt

Spelt words consist of the word's characters broken up by space for the boundaries of spelling, such that spelling in syllables is also possible to be represented.

### 4.2.21 pvc

One of the interpretative specialities of VOICE transcription is the mark-up relating to pronunciation, variation and coinages. The content model is a reduced `utterance_part` based model. For a discussion of the methodology behind PVCs in VOICE, see *VOICE Recording-Methodological Challenges in the Compilation of a Corpus of Spoken ELF* (Breiteneder, Pitzl, Majewski & Klimpfinger 2006). Where the meaning is accessible to the transcriber, a translation can be provided with the optional `desc_trail` component.

### 4.2.22 year

Years are the only numbers within the VOICE transcription format that may be transcribed numerically. Years are required to be in the full four digit format.

### 4.2.23 pause

Pauses in VOICE may be distinguished in two basic types of pause. First, brief pauses are indicated with a single full stop in parentheses. Secondly, timed pauses are encoded with the duration of the pause in full seconds in parentheses.

### 4.2.24 lengthening

Lengthening of individual sounds within words is indicated with a colon following the lengthened sound.

### 4.2.25 uncertain

Uncertain speech is encoded within parentheses. Even though the syntax of the uncertain component is very similar to the pause it is unambiguously distinguishable as uncertain speech does not contain digits.

### 4.2.26 unintelligible

The representation of unintelligible language is similar to uncertain language, apart from the practice of representing the language. The approximation of syllables is replaced by "x" characters. Unintelligible components can contain several syllables.

### 4.2.27 unint_syl

This component represents the content model of the unintelligible component. The model is based on a reduced `utterance_part` model, such that overlaps, pauses and overlaps may occur within the model.

### 4.2.28 overlap

The overlap component is used to represent and encode mark-up that is temporally overlapping with another speaker's contribution. Therefore, the content model for overlap is the same as for the utterance as such. The formal definition of the transcription format does not restrict the use of overlap, therefore any implementation like the VOICE parser has to ensure that tuples of overlaps match. The general rule is that overlaps are numbered consecutively. The numbering is conventionally restarted with `<1>` when there has not been an overlap for a longer stretch of utterances. It is possible to re-start with `<1>` when there has been at least one `utterance` without overlap. Furthermore, a count of `utterance` annotations (i.e. 6) has been defined from which it is regarded to be safe to re-start the with `<1>`.

### 4.2.29 word_char_x and word_char_X

The character content that is valid within parts of an utterance that are encoded as being unintelligible. The small letter "x" is used for normal speech while the uppercase letter "X" is used whenever a strong emphasis is noticed by the transcriber. This distinction is possible even though the transcriber does not understand the actual language. Therefore it relates to extraordinary stress or intensity recognisable in the voice of a speaker.

### 4.2.30 other_cont

Other-continuations may be indicated at the beginning or end of a given utterance. If an utterance begins with an `other_cont` component it is required that the temporally preceeding utterance ends with a corresponding `other_cont` component. Conversely, if an utterance ends with an `other_cont` component it is required that the temporally following utterance ends with a corresponding `other_cont` component.

### 4.2.31 onomatopoeic

Language that is recognised by the transcriber as being onomatopoeic. The interpretation of produced language as being onomatopoeic depends on the context of the utterance and involves a considerable degree of interpretation. The sound the speaker produces is rendered in a selection of characters from the International Phonetic Alphabet. See *4.2.35. ipa (p.42)* for a description of the used character-set.

### 4.2.32 rise

Exceptionally strong rising intonation is transcribed with a question-mark. In VOICE only intonation that is interpreted as exceptional in relation to the respective speaker's usual pitch range is transcribed.

### 4.2.33 fall

Exceptionally strong falling intonation is transcribed with a full stop. In VOICE only intonation that is interpreted as exceptional in relation to the respective speaker's usual pitch range is transcribed.

### 4.2.34 foreign

Any switch into a different language other than English is transcribed with the foreign component. In the foreign component, the speaker's position to the language is indicated. "L1" is a first language of the speaker. "LN" is any other language the speaker uses that is not a first language of the speaker. "LQ" refers to any language the speaker uses where it cannot be ascertained whether the language is the speaker's first language or not.

### 4.2.35 ipa

The `ipa` component is used to provide a supplementary phonemic transcription. The content model allows the basic latin characters plus the characters from the IPA UNICODE block (cf. Unicode Consortium 2010) and some other characters commonly used by the international phonetic alphabet. One of the letters that is neither part of the latin alphabet nor explicitly in the IPA UNICODE block, is the CARET-character "ʌ" (see appendix *PRONUNCIATION VARIATIONS & COINAGES p.223*).

### 4.2.36 speaking_mode

The `speaking_mode` component marks a portion of speech that is uttered in a specific and distinguishable way. For a list of possible values for `mode`, see *4.2.37. mode (p.42)*.

### 4.2.37 mode

This component is an umbrella for the possible speaking modes. Common modes include "fast", "soft", "slow", "loud", "whispering", "sighing", "singing", "yawning", "reading", "reading aloud", "on phone", "imitating". Furthermore, this list may be extended by the addition of other primitives. Care has to be taken that the primitives are chosen, such that the modes do not resemble other mark-up.

### 4.2.38 laughingly

This component is equivalent to a speaking mode, but due to its frequent occurrence is encoded with the shorter at-character "@".

### 4.2.39 speaker_noise

This component encodes any voiced or unvoiced noise the speaker produces as part of speech. Noises other persons, apart from the current speaker, produce are usually not transcribed, as this would render the transcripts too verbose and more difficult to read. Noises encoded with speaker noise may, but do not have to have a pragmatic function.

The optional `timed` component can be used when the duration of the noise appears to be significant for the interpretation of the speaker_noise.

### 4.2.40 noise

This component represents the open list of possible speaker_noises. Note that any additions to this list have to be carefully selected so that conflict with other VOICE mark-up is prevented.

### 4.2.41 directed

When a speaker directs some part of the conversation to a specific speaker, this is indicated by the use of the directed component. This component is a wrapper around the utterance_part component, such that any mark-up permissible within an utterance can also be used within a stretch of directed speech. The speaker_id of the addressed speaker is given in the start and end tag of the directed component.

### 4.2.42 medium_start, medium_end, medium_disc, medium_track, medium_time

These components form the boundaries of the medium component and arrange the necessary information relating to the position of the source audio medium.

### 4.2.43 gap

Gaps in transcription are indicated by the use of the `gap` component. This element is defined by a time indicating the temporal extent of the non-transcribed source and a description indicating the reasons for omission.

### 4.2.44 nrec

The semantics of the nrec component is equivalent to the gap component. It differs in so far as part of the description may be ommitted. The use of the nrec component already indicates the reason for omission, namely the lack of transcribable audio material due to not being recorded (nrec).

### 4.2.45 long_time

This component is used to encode a timespan in six digit format. The digits refer in colon-separated pairs to hours, minutes and seconds.

### 4.2.46 short_time

This component is used to encode a timestamp in four digit format. The digits refer in colon-separated pairs to minutes and seconds.

### 4.2.47 desc_trail

This content model is used for supplemented information. It is applied in contextual notes, translations and the indication of reasons for gap and nrec components.

### 4.2.48 noteref

References corresponding to notes within the transcriber notes. This component can be put at any place within the transcript body to satisfy the need to comment on arbitrary aspects of the transcription.

### 4.2.49 hash

Refers to a note concerning the transcription process.

### 4.2.50 excl

This component refers to a note concerning some interpretative aspect of the transcription, especially some indication to locate an area that might be of possible interest in later analysis.

### 4.2.51 transcriber_notes

This component is a container for collecting notes that are referenced within the text of a transcript. Additionally it is possible to add a general description of the transcribed speech-event.

### 4.2.52 general_note

A general note contains a general description. This can be either the content of a note or at the beginning of the `transcriber_notes` the brief description of a speech event. This may include contextual information to make it easier to evaluate the value of a text for further research.

### 4.2.53 hash_note

A note referring to the transcription process. Examples are unclear features or mark-up that might be ambiguous.

### 4.2.54 excl_note

This type of note component is used to indicate features that might be interesting for later research.

### 4.2.55 alpha_num

This component is mainly used in annotations and represents a mixture of alphabetical Latin characters and numbers.

### 4.2.56 alpha_char

This component comprises upper and lowercase characters.

### 4.2.57 alpha_lchar

Lower-case characters are used to lexically transcribe parts of words or whole words.

### 4.2.58 alpha_uchar

Upper-case characters are used like lower-case characters, but indicate a strong prominence of the uttered sounds that go beyond usual stress-patterns. Usually, this refers to the loudness of the produced sounds.

### 4.2.59 date

Dates in VOICE transcripts are all encoded using the eight digit short date format according to ISO 8601 (cf. International Organisation for Standardisation (ISO) 2004). This means that the first four digits indicate the year, the fifth and sixth the month and the remaining two the day of the month.

### 4.2.60 num_char

The characters used to encodes numbers. All digits from "0" to "9" are used.

# 5 The VOICE Meta-data Database

The most important aspect of a linguistic resource is the linguistic data that establishes this resource. From the perspective of the resource creation this is a straightforward assumption. The focus of a linguistic corpus lies on the linguistic aspects of the data. For VOICE this is the transcribed and annotated spoken language of the participants. Therefore, the question is why additional data, describing this linguistic primary data, should be made available. Furthermore it is a good and valid question whether the effort put into the collection and subsequently, the publication of meta-data is a worth-while endeavour. I will argue in the following for the importance of meta-data and why this kind of data is essential for research on the primary data and the key to making this primary data relevant.

There is a number of reasons why meta-data represent a value in itself. The most important reasons relate to the requirements for a sound interpretation of the linguistic data. For this, meta-data establishes and provides or re-evokes a context. This context in itself is not comprehensive, but has a limited scope. This means that only a careful selection of information is available to the researcher using the resource. While this information is clearly important to any researcher working with the primary data, it is even more important if the researcher has not been involved in the creation of the corpus. For a publicly available resource like VOICE, this has to be assumed to be the default case. Thus, first of all, it is

important to define the target audience of the corpus. The VOICE Project's declared goal has from the beginning of the project been to create a basis for a

> large-scale and in-depth linguistic description of this most common contemporary use of English by providing a corpus of spoken ELF interactions which will be accessible to linguistic researchers all over the world. (VOICE Project 2008)

Therefore, the resource has been designed to be as independent on the internal procedural knowledge of the VOICE team as possible. Meta-data, apart from documentation, is a main aspect of VOICE that caters for this particular requirement. This requirement that would not be as important for a private (i.e. unpublished) resource that has only been created for a single study.

As noted above, the scope of meta-data has to be limited in practice. Therefore, not everything on the circumstances of a speech event can be deduced from the meta-data. Thus, meta-data that is made available, has to be carefully selected and should not omit information likely to be essential to the expected types of research. Aspects meta-data might address are, for example, answers to questions of the following kind:

- Where was the text recorded?
- What is the background of the speakers?
- What is the relation between the speakers?
- How long was the event?
- What is the classification of the text within the corpus?
- Who was involved in the creation of the resource?

Answering these questions and giving information similar to these helps to identify and understand the role of a piece of data in the corpus. Multiple facets of the individual samples become accessible and henceforth a possible parameter that can be used in research. Above questions indicate some of the available facets in VOICE. Therefore, a resource with rich meta-data can answer research questions much more reliably and help to estimate and argue the suitability for particular research questions. If the resource is seen as an image of a given reality, the meta-data helps to make and support inferences about the part of reality that may be visible and observable with the material sampled in a corpus.

## 5.1 Purpose of the Database

The creation of a corpus resource involves different types of data beyond the data that is immediately related to the linguistic resource. Thus, the purpose of a meta-data database is to capture, maintain and make use of this data. The primary data, that is the linguistic data that has to be available at the end of the preparation process, is prepared in text-based transcripts. At least some of the secondary data, the meta-data, is required for the

coordination of the corpus compilation process. Therefore, it has to be available while the corpus resource is created and, thus, independently before the corpus is built and can be queried. For this reason, a relational database was set up for the VOICE project to store and make this data accessible to the project team. The benefit of decoupling the meta-data from the transcripts is that a database can be queried to measure the progress of the compilation progress and important metrics, such as the current size, the current state of the corpus balance or the distribution of properties can be closely monitored.

The data kept in the meta-data database falls into distinct categories. First of all, data with a direct connection to the collected language data is data that directly relates to the circumstances of data-collection and the participants of an interaction. Secondly, data that documents the processing of the data from collection through transcription to the eventual publication of the resource. This involves not only the documentation of many individual steps, but also the coordination of work by several individual researchers. As a side-effect, data that captures the status of the process and preparation of a resource is produced. Obviously, both, the event-related data and the data that refers to the preparation process can be a valuable resource in the generation and evaluation of research findings. But, not all meta-data that is maintained to coordinate the efforts during the preparation process is relevant to an external researcher.

Even though meta-data is not directly the main objective of the resource and this kind of data is also not an inherent part of the language data as such but secondary to it, it provides additional information that is directly linked to the original data. This meta-data is essential to come to a legitimate interpretation of the data. Data with this kind of relation to the primary data is, in information processing, generally called meta-data as it is data that is descriptive of some other data. For language data this means that the meta-data can be descriptive of aspects of the captured data. For a spoken language corpus like VOICE this includes for example information on the event, the setting, the interactants. Furthermore, it comprises information on the used recording equipment and procedural information. This is information on the different stages and responsibilities during the transcription and the quality assurance processes of the project.

Above examples indicate that meta-data itself is a very heterogeneous category. It ranges from data that almost immediately relates to the language data, as for example an indication of the speakers' first language (L1), to data that is documentary of the process and the decisions that were taken during corpus creation. This implies that the different types of meta-data are created at different points in time as well as for a different purpose. While, in the case of VOICE, meta-data that relates to the individual speakers of an event was actively collected during the field-recordings, the meta-data that documents the structure and the creation of the resource was generated and captured during the preparation of the resource.

## 5.2 Structure

The structure of the meta-data database is centred around the notion of the 'speech event'. That means that any meta-data that is collected is directly or indirectly related to a speech event. It is important to note that this central relation also includes meta-data related to the processing of the resource. This is mostly due to VOICE's declared aim to capture complete speech events and use these complete events as sampling unit in the corpus. Such a fundamental methodological decision has to be also reflected in the preparation process of the material as well as in the kind of information that is collected. Hence this is not only reflected in the kind, but also in the structure of the data in the meta-data database. For VOICE, this means that the central point of reference to any information is the speech event. Thus, any information in the database is eventually related to a particular speech event. Most of the information is directly related to the speech event, while fewer items are related via a level of indirection, such as information on speakers.

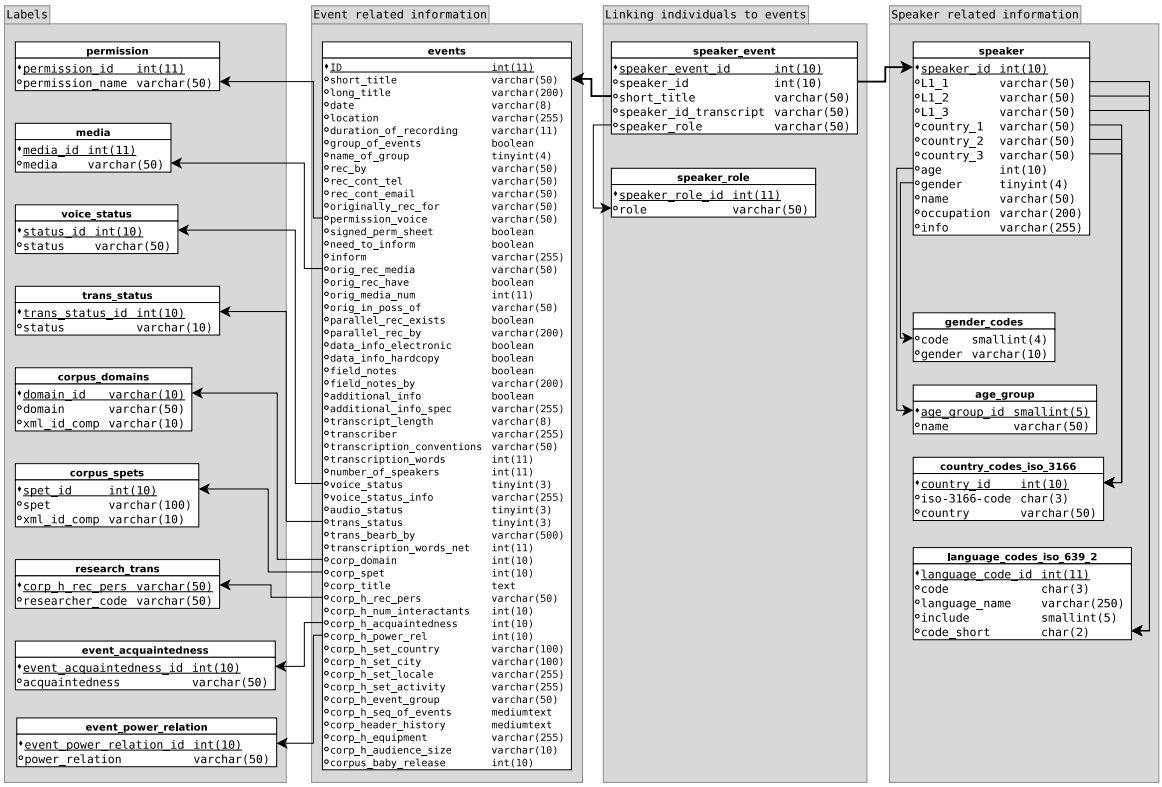



Figure 17: Schema of the VOICE meta-data and administrative database. The column on the left groups tables with descriptive labels. The second from the left is the central part of the database, as it is the table with the core information on the event. The third column defines the relation between individual speakers in the fourth column with the event related information in the second column.

The overview in *figure 17* outlines the basic structure of the database. Each of the shaded columns is related to a particular area. The light coloured boxes are descriptive of a particular table available in the database. The fields of a table are defined by the individual lines in the box assigning a field-name to a data-type. The information in the indivudal

tables of the data-base is related, hence the term 'relational database' that is used for this type of database product. The relations between the tables and their direction is indicated in *figure 17* by the arrows pointing from a source field of one table to a destination field of a different table. Whenever the fields of both tables hold an equal value, the records – that means the line in the table – including all columns are related. In fact, simplifying here, one could assume that the fields of both tables related in this way form a wider virtual table with the fields of all tables combined. In my work with researchers without a computer-science background, I found that this image helped with the understanding of the concept of relational databases. Nevertheless, it is important to stress the advantages of relational databases that are hidden beyond this simplification. This includes advantages like the possibility to relate as many records from one table with a single record of another table. Using the image of a virtual table, this would be equal to a table with variable width, depending on how much information is required for a specific record. Constructs like this would not be feasible and error-prone to maintain in a tabular form, such as for example with Excel files. It is precisely the advantage of this flexible format to be safe and simple to define within relational databases.

The relations in VOICE's meta-data database fall into three categories of types of Information. That is information that can

- occur exactly once per event.
- occur exactly once per event but is restricted to a closed set of predefined values.
- may occur several times in one or more events.

These different kinds of relation are expressed in distinct ways. Information that occurs exactly once per event, such as for example the 'short_title', the 'date' or the 'location' of the event are directly part of the table 'events'. This is the most appropriate way to encode this kind of information as the piece of information needs a fixed size, that means one column in the table, and can hold a wide range of different values. The second type, that is information that occurs exactly once, but is restricted to a set of predefined values, is implemented via a reference to a record in a different table where these values are defined. Instances of this type of data are the assignments of 'corpus_domains' or 'corpus_spets', which represent the stratification of the corpus in domains and speech event types. Obviously, this data is constrained to the values available in the specific corpus design. The third type of relation is the most flexible, but also the most complex relation. It allows, for example, the assignment of several events to several individuals[24]. This linkage is established via a table 'speaker_event' where each row event can hold references to a speech event and a speaker. Any speaker and any event can be linked by an arbitrary and

[24]In VOICE individuals are persons who occur as speakers in speech events. Therefore, especially when a group in a series of speech events has been recorded, it is possible to relate several speech events to individuals and vice-versa.

therefore unlimited number of records in this table. Note that the role of the speaker in the event is defined in this table, as the speaker's role is neither a property of the speaker as an individual, nor the event, but a property of the relation of both. As this relation is expressed in the table 'speaker_event', this table is also used to establish a further relation to a value of the closed set for speaker roles. VOICE's meta-data database uses these three types of relation to express the different relational cardinalities needed to model the available data. The term 'cardinality' expresses how many items are related to how many other items in other tables. Most relations for VOICE have the maximum cardinality $n : 1$[25]. Hence, each record at one table, an event, is related to exactly one record in the related table. The exception is the relation between individuals and speech events which has the cardinality $n : n$. That means each record at one table, an event, is related to one or more records in the related table, the individual speaker.

## 5.3 Deployment

The database for VOICE is set up with the database server MySQL, which is open-source software and thus free of any license costs. The database server is available from within the project's shared virtual private network (VPN). That means that the server is available to any project member with a working internet connection who is granted access to the project's network infrastructure. Therefore, the database can be securely and simultaneously accessed by different persons from different computers. This is especially important as several of the project members had to work with the data at the same time.

## 5.4 Using the Database

Having a defined structure for the collection of data is one aspect of the design of a database. The other side, which is in practice as important as the structure of the data itself is the provision of interfaces and the integration of the data-base in a project's work-flow. VOICE has two main access scenarios for the database. First of all, it is the central point for the collection and maintenance of event related meta-data. Secondly, this data is used by programs in the course of the preparation of the XML resources from the transcripts. This means that the database has to provide interfaces for human agents as well as for programs. The interface for the human agent is particularly important as consistent data-entry is an ultimate prerequisite for the latter publication of the meta-data. Thus, this interface is described in the following section.

[25]The constraint in this notation expresses the maximum cardinality, that means how many items of a particular entity can be involved in the relation. The minimum cardinality, that denotes conversely the number of times an entity has to be involved in a relation. For the meta-data database used in VOICE only the upper boundary is set, such that a minimum is not required. For a brief introduction to the cardinality in database relations (see O'Neil & O'Neil 2001: 399ff).

### 5.4.1 Database Users

The team involved in the compilation of VOICE has access to the database server by means of an Open Document Database[26]. Open Document Databases combine a database back-end with the possibility to define simple front-end user interface built on forms for data entry. Furthermore it provides a user-interface that guides the user through the creation of queries and printable reporting documents. These facilities proved to be especially useful to continuously control and steer the progress of the transcription process.

The user-interface mirrors the data-base structure (see *5.2. Structure p.48*) in so far as the central and most important concept is the speech event (see *figure 18*). Furthermore, forms are provided for the entry of speaker information (see *figure 20*) and additional meta-data that is required for the later publication of the corpus texts as TEI XML documents (see *figure 19*). The complexity of the relational linking of records of different tables (see *5.2. Structure p.48*) is handled by the interface and hidden to the user. Data of which the entry is restricted to a set of values defined in another table of the database is displayed as choices in drop-down boxes, for example in *figure 18* the entry with the label 'Domain', 'Voice Permission' or the type of the 'Original Medium'. The user of the database interface does not need to know about the technicalities, but can select from the predefined values. In this way the user-interface is built to mirror the structure of the meta-data for the project. Nevertheless, the complexity of the database structure is reduced and at least partially hidden by the user-interface.

In the following, the forms for data-entry are briefly illustrated and introduced. As the graphics are taken from real data sets some of the information is blurred to keep some sensitive of the information confidential. This is to protect the personal rights of those involved in the gathering, the preparation of the transcripts and of course the individual speakers. This information is mostly personal information or information from which it could be possible to guess the identity of an individual. This kind of information, such as the contact information of the person who recorded the data, is in the meta-data database not for the purpose of publication, but as data that supports the preparation of the resource. This distinction again emphasises the twofold importance of meta-data, both as information gathered in the database for the eventual publication and information that is needed for the internal coordination of the corpus building effort.

#### 5.4.1.1 Main Event Data

The data entry for the main event in *figure 18* is centered around the recording, the transcription and the suitability for the inclusion in the VOICE corpus. Thus, the data that is entered in this form is most important for the administration of the corpus compilation process. For this reason, this form represents the initial incentive for keeping information

[26]The Open Document Format is a standardised open format and an implementation has been provided by the OpenOffice.org suite of desktop applications (see Oracle 2011).

on the event in a database and, hence, is centered on the administrative and not on the publication aspect of the corpus compilation. For the editing of the latter, the form in *figure 19* is used.

On the left hand side of *figure 18* the first box 'Core Information' groups information central to the event. This is information like a descriptive title of an event, the date of recording, the duration and the place where it has been recorded. The second box 'Voice Status' denotes the current status for VOICE. A selection of 'Corpus Baby' indicates the inclusion in the corpus, while the other indicate that it might be suitable, is 'pending' or will be discarded. This distinction is important as the project gathered far more recordings than were eventually selected for inclusion. The following box 'Recording' groups information on the recording, including not only information on the collected audio data, but also whether important supplementary artifacts like the 'Data Info Sheets'[27] are available. Moreover, on the right hand side, three boxes describing properties of the transcript are available. First of all, the 'Transcript Status' which is indicative of the quality of the transcripts and ranges from values for 'no transcript available' to 'double-checked (C2)'. This information is essential, together with the 'Transcript Characteristics', to monitor the progress of the corpus compilation. The status of the necessary pre-processing of the audio material is described in the box 'Audio Status' and ranges from the file not being available as a computer file to being prepared for transcription. This information is required to estimate how many events are prepared to be handed to the transcribers, such that the project could use the workforce of external transcribers most effectively. The last box in the right column is dedicated to additional information on the event and the legal status. The most important aspect here is that the participants' permission for publication has to be available prior to the inclusion in VOICE. Finally, the box at the bottom of the form indicates the text classification, that is its position in the stratification of VOICE. This position is determined by the selection of the domain and the speech event type.

#### 5.4.1.2 Event Data for Publication

The second part of the user-interface that is specifically concerned with information on the level of a particular speech event is the form in *figure 19*. In contrast to the previously described form, this form is focused on the information that is to be published as meta-data in the header of the corpus texts. That means it is not used for administrative purposes, but for the assembly and controlling of the meta-data for publication. Some of the data is not exclusively used in this form, but shared with the form in *figure 18*. The shared data is write-protected in this form to prevent accidental change of core event data. This state is indicated by the display in slightly shaded text-fields. The shared fields are the domain, the

[27] Data info sheets are preset forms filled in by the recording personnel and furthermore supplemented with notes like seating plans, indications on the assignment of speaker sigles and other similar material that is useful to the transcribers.

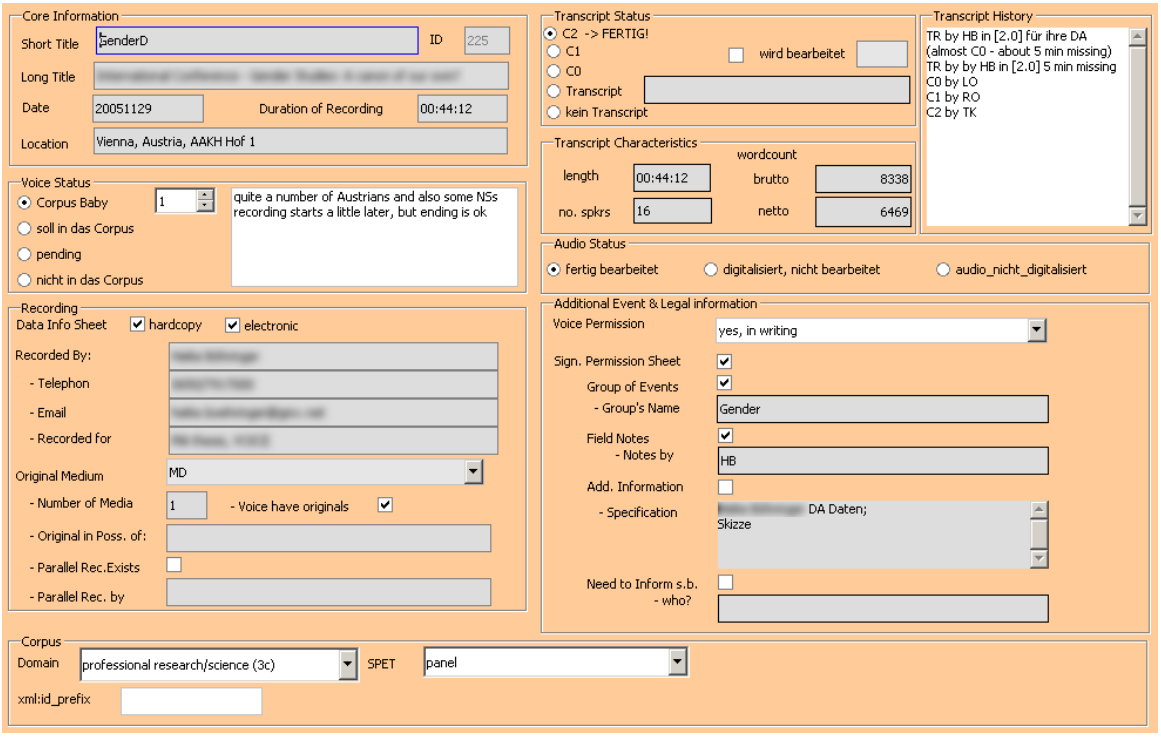



Figure 18: User Interface for the entry and editing of event related information.

speech event type, the duration of the transcribed recording, the date of the event and the number of speakers.

As the information entered via this form is included in the header of the final corpus format, the labels of the boxes used for grouping information are the names of the corresponding elements of the TEI header. That is, the recording statement 'recordingStmt', the participant description 'particDesc' and the part that is dedicated to notes on the event 'notesStmt'. It is important to note that the meta-information given here is not exhaustive and obviously misses aspects like for example the description of the stages and dates of completion in the revision description 'revisionDesc' that can be directly inferred from the individual transcripts.

#### 5.4.1.3 Speaker Information

Information on individual speakers plays a very important role in VOICE. As VOICE comprises mainly field recordings in diverse settings in several different countries, the core properties of the speakers vary greatly. Core properties are, for example, the speakers' language background, their role in the speech event, their occupation and their age group. This information is mainly interesting for qualitative research as for many aspects, such as language background, the total numbers of speakers and hence the number of uttered words vary greatly between first languages or groups of first languages (cf. VOICE Project 2009).

The form in *figure 20* is used for the entry of user related data in the database. The

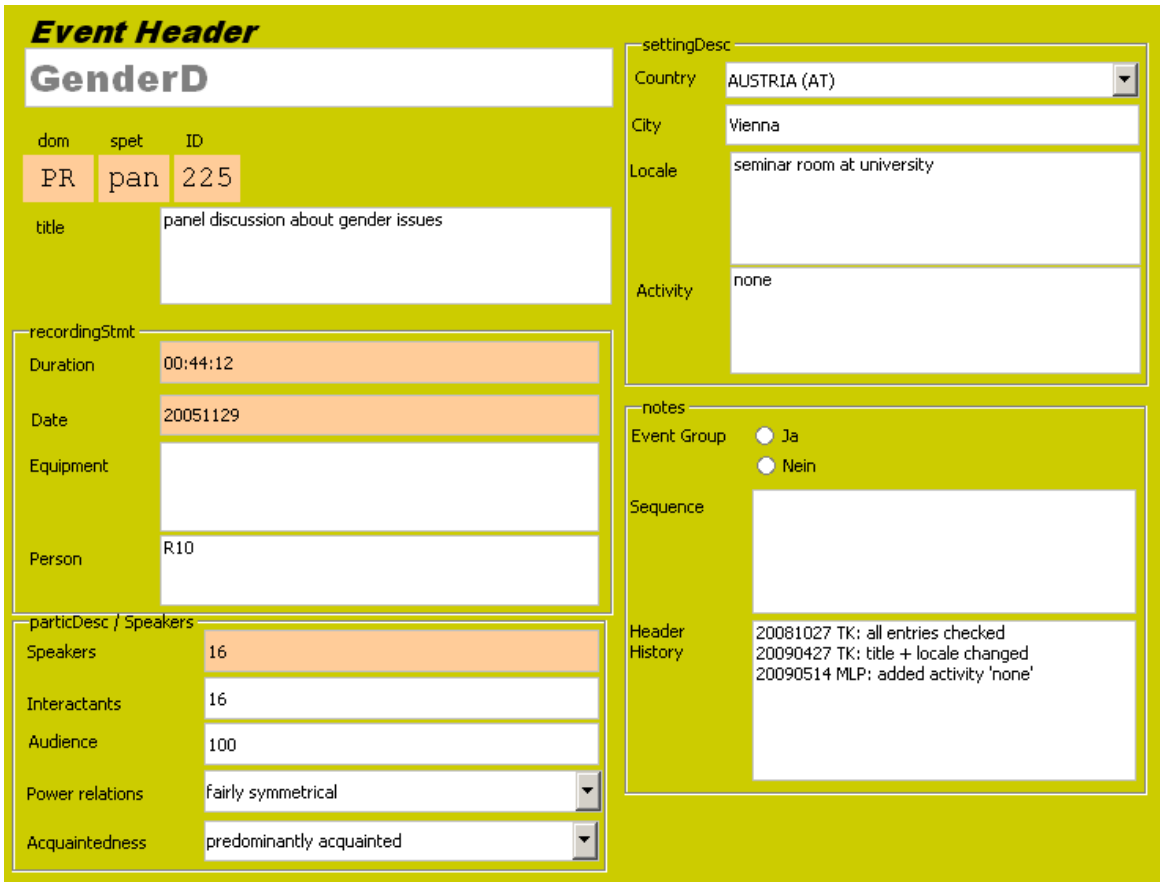



Figure 19: User Interface for the entry of event related information specific to the publication of the material as TEI-XML documents.

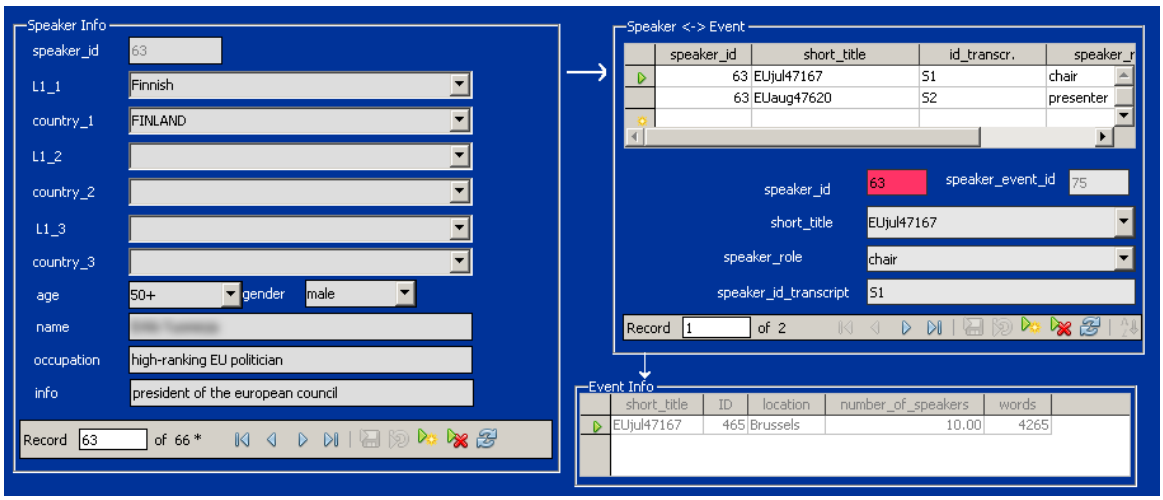



Figure 20: User Interface for the entry and editing of speaker related information.

form establishes the relation between the individual speaker and a specific speaker in the speech event. VOICE distinguishes between 'individuals' and 'speakers'. 'Individuals' in VOICE refers to 'individual speakers', but for the sake of brevity and to prevent the two distinct properties to be confused it was decided to use the shorter 'individual'/'speaker' dichotomy. However, note that the table names in the database schema (see *figure 17*) do not reflect this decision. This is due to the fact that the database schema was designed before the necessity to have clear and distinct terms for both concepts became apparent.

The left part of *figure 20* is dedicated to the entry of the details of the individual. This part is dedicated to basic information on the speaker that is not dependent on the context of a specific speech event. Data of this kind are the first languages (L1s), the age group, name and occupation of the person. As individuals in VOICE usually appear in temporally closely related groups of speech events and VOICE is not designed for longitudinal research, these categories are stable within VOICE. That is, individual speakers do not appear in varying age-groups and occupations.

The right hand side of *figure 20* establishes the relation between the individual and the speech events a speaker appears in. Some of the speaker information in this section is descriptive of the relation of the individual and the event. Most notably the 'role' of the speaker in the speech event, and the reference to the assigned sigle the speaker is identified with in the original transcript.

### 5.4.2 Programmatic Access

The programmatic access to the database, as for example used in the VOICE Parser, is in most parts defined by the database product used for the deployment of the database schema. VOICE uses a MySQL (Oracle 2010) database. MySQL is a free open-source database product which provides access to the data with connectors using the Simple Query Language (SQL) (cf. Digital Equipment Corporation 1992). This language is an industry standard designed to select and extract data from relational databases. As MySQL is part of the default software stack[28] used for many websites, libraries and bindings are available for most common programming languages. The VOICE Parser uses the database access module of the Perl programming language to aggregate the data for individual speech events. This data is then injected into the framework of the used TEI XML format.

# 6 Corpus Publication Format

This section will present the corpus encoding format of VOICE. Especially, users of VOICE who are planning to use VOICE XML will learn in which ways the corpus format reflects the annotation as specified by the VOICE transcription format (see *4.*

[28]The combination of the operating system Linux with the web-server Apache, the database MySQL and traditionally either of the script-processors Perl, PHP or Python is commonly referred to as the LAMP stack. It is one of the most commonly used software stacks for serving web-sites. (cf. Lee & Ware 2003)

*Redesigned Transcription Format p.29*). The rationale behind defining such a format, that is, a format distinct from the mark-up that is used in the process of transcription will be explained. Furthermore, approaches inherently different from the encoding of VOICE will be presented based on their historic importance and the basic assumptions governing the methodology. This part will therefore present an overview of different fundamental concepts, such that it will be possible to follow and understand the choices made for VOICE. Apart from these general decisions, details on the chosen TEI based format, extensions of this format and the representation of data representing the corpus structure, will be discussed.

## 6.1 History of text encoding formats

The use of collections of texts for linguistic enquiry, especially written texts, significantly predates the use of computers. The systematic use of larger text collections became popular with the larger processing capacities of computer-based search-facilities. Therefore, using authentic language data in larger text collections became recognised as a distinct set of methods for linguistic enquiry which was later labelled 'corpus linguistics'.

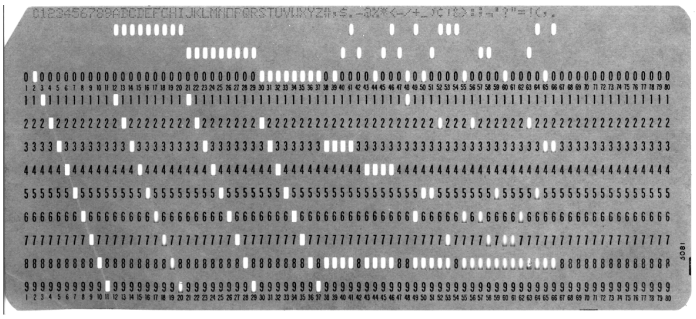
One of the first widely recognised corpus resources is the *Brown University Standard Corpus of Present-Day American English* (cf. Francis & Kucera 1979). The Brown Corpus was compiled in the early 60s. For the time, the size of the corpus was extraordinary, as it comprised more than a million words. Especially when comparing to contemporary corpora of written English, as the *American National Corpus* (cf. American National Corpus Project 2009), or the *British National Corpus* (cf. Oxford University Computing Services 2009) the size of the Brown Corpus is modest. But it was nonetheless a remarkable achievement, as in the early 1960s even the transfer of written language into a computer system was a tedious task including hours of manual work preparing punch cards for the input reader. How much of an achievement the Brown Corpus actually was, becomes evident from the fact that it has been one of the most quoted corpora for decades and is still actively used[29].

Not only the scientific impact of the Brown corpus, but also the encoding decisions make it a worthwhile starting point, when looking at corpus encoding. In the original version of the Brown corpus, there is not yet a layer of grammatical annotation. Even though Brown does not encode spoken language, most of the design decisions concerning the actual technical representation of the language data are analogous. Brown is focused on written discourse. Fiction, usually including a fair amount of dialogue, is only admitted in Brown when the ratio of dialogue is less than fifty percent (cf. Francis & Kucera 1979). As Brown includes no authentic spoken language data, there is no mark-up specifically dedicated for features prominent in spoken language. The choice of language samples leads to the choice

[29]Bibliographic searches show that every year hundreds of papers still use the Brown Corpus. The use of the Brown Corpus ranges from training material for models in computational linguistics to its use as a point of reference for comparision.

of mark-up when it relates to features of spoken language as it is represented in fiction. Even though Brown does not represent spoken language, there are aspects of encoding that are inherent to the encoding of language, spoken or written. This is especially true in areas where the encoding of language follows the written mode, or, as one can argue that written language itself is loosely modelled after spoken language. One area where there are certainly similarities between many encoding schemes and written language and written spoken language is the representation of temporal aspects. Especially, as far as the representation of language as a sequence of lexical items is concerned, the Brown corpus is an important project, which became a reference-point followed and contrasted to the encoding schemes of later corpus projects. The Brown corpus marks the beginning of a new way of working with textual data.

The question of the implementation is theoretically independent from the design principles governing the design process of a corpus project. Nonetheless, the implementation has to be both feasible and reflect the design decisions. Usually, this leads to a compromise that shows in the format. For the Brown corpus, this is reflected for example in the encoding format in such a way that ensures that each line of the encoded text fits on 80 character cardboard punch-cards. Brown encodes the texts as sets of lines prepended by a number



Source: http://commons.wikimedia.org/wiki/File:Blue-punch-card-front-horiz.png

Figure 21: IBM-compatible 80 character punch-card. The characters are encoded by a combination of numbered punch-holes.

uniquely identifying the individual line of text. The numbering of the lines represents the position of the line in the text. Thus, a simple sort algorithm is sufficient to reconstruct the lines from the punch-cards, even when the punch-cards may accidentally not be in the right order. But, as encoding errors happen and it might be required to insert text at a given point, the numbering scheme skips several counters for each line (See *figure 22*). At a later time, if there were additions to the text, some lines could be introduced without having to renumber all punch-cards (cf. Francis & Kucera 1979). On a different medium, like a floppy disk or hard-disk, this renumbering scheme would not be necessary. Here it becomes obvious that at least some of the choices concerning the format of the corpus are governed by the used technologies. Others, like the encoding of annotation, might follow a

```
A03 1350 schooling costs.  #CONTRIBUTIONS TO SCHOOLS# The scholarship plan
A03 1355 would
A03 1360 provide federal contributions to each medical and dental school equal
A03 1370 to $1,500 a year for one-fourth of the first year students.  The
```

Figure 22: Example from Brown1_A.TXT with a correction on line 1355

different rationale, but, as the implementation has to be designed to work with the available technology, it has to combine the scientifically desirable with the technologically possible. The encoding decisions have to match, at least to a certain extent, the confines of the feasible and possible. Nevertheless, challenging these limits with technology-external requirements can lead to a development in the applied technologies. An example of developments driven by technology-external requirements is the rise of the 'Standard Generalized Markup Language' (SGML) (cf. International Organisation for Standardisation (ISO) 1986).

### 6.1.1 Pre-SGML formats

Taking the Brown Corpus as a starting point, it is important to note that even without established predecessors it was an aim of the compilers to use an existing technology. That means, a technology that has been proven to work for similar tasks, even in a different context. When Brown was compiled, there was next to no experience in text encoding for scientific purposes. Thus, the compilers extended a pre-existing standard which was not designed for research, but was used for the encoding of written documents. They used the *Notation System for Transliterating Technical and Scientific Text for Use in Data Processing Systems* (cf. Francis & Kucera 1979) as a foundation for their encoding standard. Most areas where the encoding had to be extended were located in the texts dealing with fiction.

Even though many of the concepts present in other work could be reused, the resources differed to a great extent, as there has not yet evolved a common best practise in respect to the encoding format. Therefore, every project creating a resource had to build their own set of tools to make the texts accessible. For Brown this led to the publication of several concurrent encodings. One with the mark-up and another version just containing the text. Had there been a standardised set of encoding techniques available, as the definition of the SGML standard provides, this would have been different. For this reason, it is valuable to distinguish between pre-SGML and SGML encodings, where the today more popular Extensible Markup Language (XML) (Bray, Paoli, Maler & Yergeau 2008) is an application of SGML.

The fact that the implementation of the pre-SGML encodings is usually a unique endeavour that usually has not too many neighbouring projects with compatible encodings has a strong impact on the implementations of the tools a researcher can use to access the data. Even though the benefits of following standards usually outweigh the benefits

a project can gain from having a custom non-standard encoding, it is worth seeing the benefits of such an encoding. These benefits explain to a large extent why projects at a certain time were likely to choose a custom encoding format. Furthermore, the intended use and hence the purpose of a given language resource changed during the history of corpus encoding. These shifts also changed the weight of some benefits or drawbacks of encoding-choices. For example, in the 1960s it has not been as important as today to build a resource that is usable on a wide range of operating systems or a wide range of different computers. At that time a language resource was likely to be used on mainframe computers. This implies that it is neither necessary to create the resource to be transferable to a different machine or type of machine. Researchers who would like to have access to the resource, would have to have a terminal-login and do their research with the tools available on the mainframe computer. Thus, neither the language resource nor the tools used to work with the data had to be portable between different types of machines. But, as the machines were much less powerful, it was an essential requirement to do the computational tasks as efficiently as possible. Computers have rapidly developed during the last decades, most significantly in the amount of available memory, data-storage and processing speed. All these grew exponentially, such that factors that once where limiting, became optional aspects for the optimisation of a resource. Computational inefficiencies that are acceptable today, would have rendered a program unusable at that time. Consequently, the structure of encoding formats was, much more than today, determined by the limitations of the existing computers. But, as sharing data with other projects and using data on different kinds of computers was at that time not such an important rationale, the tools as well as the encoding of a language resource could be tailored to make the most of the limited resources for a specific purpose.

There are three areas where the resources could be limited and where strategies to match the implementations with the available resources have to be employed. First of all, the persistent memory for keeping the resource available to the processing machine. Secondly, the amount of memory that is available on a given machine for the processing of the resource and for calculating results. And, the last important factor is the processing power. This, here, refers to the amount of subsequent operations a machine can perform in a given amount of time.

Persistent memory is the memory used to keep and store data for the use in informational systems. Persistent memory is any kind of memory that stores the information independently from the power-state of a machine. That means, the information stored in persistent memory is kept between power-cycles and is, if write-protected relatively safe from errors malfunctioning software could cause. From this perspective it is important to keep at least a copy of precious data in persistent memory. Persistent memory, due to the used technologies, like electromechanical spinning hard-disks or the earlier punch-cards, is in respect to the read/write performance a bottleneck in the operation. Only the very

recent development of solid-state disks slowly closes the gap between the performance of persistent and non-persistent memory. Still, solid state disks are by magnitudes slower than access to random access memory (RAM). For this reason, persistent memory is used to store data and programs, while they are loaded for faster processing into RAM when they are used. As persistent memory systems are the parts of informational systems that keep the data for long time, the properties of these memory systems influence the choice of encoding the language resource. Early systems were, first of all, very expensive. As every resource that is kept in persistent memory is permanently using the space that is allocated there, resources in persistent memory cause fixed costs dependent on the size of the resource. Therefore, to keep the costs low, early systems were designed to use as little persistent memory as possible. The data format of Brown, for example, therefore uses very short sequences of characters for annotation. These sequences of characters represent the features the Brown Corpus encodes within the textual data. The character sequences are not chosen on grounds of mnemonic aspects establishing a link to the encoded meaning. Thus, they are hard to read and difficult to write for a human agent. This is especially important for the Brown corpus, as it was encoded directly into the corpus format on electromechanic punch-cards. The influence of the price of persistent memory almost vanished with the plummeting memory prices. The precedence of resource saving arguments relating to the use of persistent memory have lost their weight. Therefore, the benefits of more memory consuming but explicit and expressive corpus-formats outweigh the resource savings of formats optimised on small size. Furthermore, the time required to access persistent memory has much decreased and can in many cases, with the relatively modest requirements of text encoding, be ignored.

The main memory of computer-systems, the random access memory (RAM), also has had constraining impact on choices for the representation of linguistic data. As with the persistent storage of data, the size has generally been the most limiting factor. The main memory is usually very fast, compared to persistent memory. Therefore, operations on the data, like search and retrieval operations, should be performed on chunks of data that are kept in memory throughout the entire operation. Caching the data back to persistent memory would render the operations much slower and is thus generally avoided in computer systems. This idea is therefore also reflected in the structure of the corpus encoding. While today's memory-consumptive XML encodings are suitable for fitting multi-million word corpora in the available RAM of an average contemporary machine, corpora like Brown had to face far greater challenges in this area. One major difference in the encodings of Brown and TEI based corpora like VOICE is that the annotation of VOICE is held in a treelike structure, whereas Brown uses a flat structure where markers indicate the beginning and end of the annotation. Processing and using flat structures requires much less overhead and is therefore less memory-consumptive. But, more complex queries on

less complex annotation require, if at all possible, much more effort in the processing of the query.

The third area that has strong impact on the performance of computer systems is the processing power. Generally, this refers to the number of operations the processor of a computer can perform. Operations are amongst others addition, subtraction or comparison. As operations are in the domain of the programs and algorithms, the processing power has its biggest impact in this area. The encoding format of a static resource like a text corpus defines the structure of the encoded information and not the operations on it. Thus, it is not directly influenced by the exponential development of processing power over the last decades. But, as the three mentioned aspects, persistent memory, main memory and processing power are interdependent and have undergone a similarly rapid development, the processing power is frequently used to estimate the suitability of systems for specific tasks.

The pre-SGML era in humanities computing can be characterised by its striving towards efficiency above interoperability. This lead to many different encodings that generally do not share a great basis of interoperability for different encoding projects. At that time interoperability was less an issue than today, as interoperability is only required if there are comparable resources that can benefit by a joint usage. This changed later with the improving capacities of computer systems and the emergence of numerous linguistic and non-linguistic text-encoding efforts. There, the focus shifted from efficiency towards more rigid standardisation and interoperability.

### 6.1.2 SGML/XML based formats

Formats based on SGML or, more frequently today, XML based formats have become the de-facto standard in language encoding. The aim of these encodings is, as the name of the earlier Standard Generalized Markup Language (SGML) (cf. International Organisation for Standardisation (ISO) 1986) indicates, to set standards in information processing and the encoding of data. Establishing standards in the encoding and, notably, also in the processing of data became the main objective of these technologies. SGML and later XML, the Extensible Markup Language (cf. Bray, Paoli, Maler & Yergeau 2008) were rapidly adopted by implementers of language resources. This shift from efficiency towards standardised resources illustrates several important points. Mere efficiency became less important than the possibility to compare different resources in one area. This became obviously more important with the growing number and range of different resources available.

SGML and XML are used in the literature interchangeably as technologies for the encoding of data. Despite the common goals and a substantial technological overlap, there are significant differences between the two encoding formats. Nevertheless, XML is defined as an application profile of SGML. Therefore, it is derived from SGML and restricts numerous aspects of the syntax of SGML. Hence, any well-formed XML 1.0

document is at the same time a valid SGML document. The reverse is – as XML is a subset of SGML – generally not true. XML is much more restrictive in the possible document structure than SGML, without loosing its expressiveness. XML documents are a representation of elements that form, by their physical nesting, a single-rooted tree of nodes. Therefore XML documents are especially suitable for the representation of highly structured data. SGML on the contrary, appears on the surface much less structured[30]. This is mainly due to the fact that SGML allows for more shorthands, where the range of the mark-up is only deducible from the DTD. SGML for example allows for "unclosed start-tags, unclosed end-tags, empty start-tags [and] empty end-tags." (Clark 1997). Especially the unclosed tags make some documents impossible to be interpreted without the DTD, as the rules for the implicit closing of the elements are defined there. XML solves this, as it always requires start and end-tags. Tags are always explicitly and never implicitly closed in XML. Small adjustments of this kind allow for technically much simpler processing tools, without constraining the expressiveness of the mark-up. XML is, nevertheless, not just a simplified version of SGML. While formally it is precisely that, it has been extended with mechanisms for extensibility that allow for further specification. For example it is in an XML document well possible to mix different annotation schemes for different purposes in a single document. It is for example possible to use 'MathML' (Carlisle, Ion, Miner & Poppelier 2003), to encode mathematical formulae, and 'SVG' (Dahlström, Ferraiolo, Fujisawa, Jackson, Lilley, McCormack, Schepers, Watt & Dengler 2010), for the encoding of graphics, in a document which follows a schema for a text-encoding project such as a TEI based format. Even though SGML and XML are based on the same technological foundations, they can not be seen as being interchangeable.

The definition of XML as a successor of, and based on SGML demarcates a development which led in many parts of the encoding industries to a shift from SGML to XML. I would see the main reason for this shift in the movement from standardisation towards extensible standardisation. As argued above, the simplified features of XML makes it simpler for programmers to implement generic tools to process the XML encoded documents. Less has to be known or assumed about the definition of a specific XML based mark-up language, to be able to make use of the encoded data. Furthermore, the mechanisms that warrant and improve the extensibility of XML based mark-up contributed to its specific position as the successor of SGML. These mechanisms are:

- XML Schema can define a document more accurately than DTDs
- Mark-up can be defined in XML Namespaces (cf. Fallside & Walmsley 2004)

This, and the number of co-standards to XML, make XML based mark-up flexible for very diverse usage scenarios. The use-cases range from machine-to-machine communication to

[30] SGML documents may define some of their structural information in a seperate file. That is, some of the structure is not visible when looking at a specific document without referring to the document's particular Document Type Declaration (DTD) (cf. International Organisation for Standardisation (ISO) 1986).

encodings designed for the manual encoding of textual data. Important co-standards are, among others:

- XML Stylesheet Language Family (XSL), for, most importantly, the definition of transformations and the visual display (cf. Quin 2009)[31]
- XML Include, to include resources that are physically at a different location (cf. Marsh, Orchard & Veillard 2006)
- XML Query (XQuery), to query encoded documents as structured data (cf. Boag, Chamberlin, Fernández, Florescu, Robie & Jérôme 2007)

Many of the applications that use XML can be expressed by these co-standards. Therefore, less tools have to be developed outside the XML family of standards. This leads to an improved portability, particularly because many projects share the same programs to use their resources. VOICE uses all of the above-mentioned standards either in the preparation of the resource or the on-line presentation of the corpus.

## 6.2 XML as the Basis of an Encoding Standard

One of the main objectives of VOICE has been to create a language resource that may be used, analysed and processed with a wide range of linguistic tools. This also includes the possibility of using methods and tools not yet available or conceived. Hence, the encoding standard used for VOICE is required to be open and adaptable to the requirements of various infrastructural settings. Therefore, it has to be either universally accepted as an universal input standard or transformable for the particular needs of a specific infrastructure. There is unfortunately only little consent on the type of input linguistic tools accept. *WordSmith Tools* (Scott 2010), to name one, is not very good at making use of the semantics of XML based encodings, whereas it is a powerful tool to analyse plain text. Other tools, like *XAIRA (XML aware indexing and retrieval architecture)* (Dodd 2010), can make extensive use of information encoded in XML, but has only limited support for statistics (see Xiao 2006: 102ff). Therefore it is important to store the data in a manner which does not eventually prevent the researchers to use a linguistic resource with the tools they need for their particular analysis[32]. This holds especially true when annotation is to be involved in the analysis and not just the plain text. Thus, as there is no universally accepted input standard, VOICE decided to opt for a format where it is possible to transform the information to accommodate and recode the data for particular application scenarios. One meta-language that is particularly fit and even designed to define custom mark-up that fulfils the above requirements is the Extensible Markup Language (XML) (Bray, Paoli, Maler &

[31] So far the VOICE infrastructure makes only use of one of the languages specified in this language family, namely the *XSL Transformations (XSLT)* (Clark 1999). Together with XQuery it is the main building block for the online interface of VOICE (see VOICE 2009)

[32] Tools for linguists tend to handle very different input formats, e.g. text-based formats, comma-separated values, various XML schemas.

Yergeau 2008). VOICE tried to avoid the necessity to implement and maintain a complete tool-chain for defining transformations into different output formats. It became clear that XML technologies would serve our needs, from this perspective, as transformations would only have to be defined on a high level on XML co-standards like the *Extensible Stylesheet Language Family (XSL)* (cf. Quin 2009) and can be used with existing tools.

## 6.3 Different XML based corpus formats

XML is the technical foundation of many contemporary linguistic corpora. General purpose corpora, that means corpora with a wide range of linguistic data, like the British National Corpus (BNC) (cf. Oxford University Computing Services 2009), the American National Corpus (ANC) (cf. American National Corpus Project 2009), but also more specialised corpora like MICASE (cf. Michigan Corpus Linguistics 2010), VOICE (cf. VOICE 2009) or the AMI-Corpus (cf. AMI Project 2009) use XML as the foundation of their corpus encoding. The difference of the encoded features and research perspectives are reflected in the chosen XML-based encoding. This is due to the fact that XML is a meta-standard that can be used as the basis to define a particular corpus encoding format, but not an encoding format in itself. That means, even though XML is used as the basis for an encoding format, equivalent features might be expressed in different ways. Hence, the approaches towards the encoding can be very different in projects with different research perspectives. While it would be impossible to cover the whole spectrum of possibilities, as XML can express an infinite number of encodings, it is well possible to distinguish few basic and commonly used strategies for the encoding of linguistic data.

Two very distinct approaches can be identified when comparing encoding of annotations. The linear referential and the document-oriented hierarchical formats are two formally distinct extremes in their encoding approach. Even though many complex real-world encoding schemes, like VOICE, are located somewhere in a continuum between these extremes, it is worth describing them, as they are fundamental to rich annotations. Both approaches are described in the following.

### 6.3.1 Linear Referential Formats

The much recognised work on a *formal framework for linguistic annotation* (cf. Bird & Liberman 2001) formally describes annotation as a set of graphs with nodes referencing some kind of temporal structure (see discussion in *2. Annotation p.4*). This can be achieved with timestamps relating, for example, to an audio source, but also with references to locations in a text. One important aspect of this concept is that it encourages the referencing of the different annotation layers to a common baseline. Referencing is, in an abstract sense, a mechanism to use information at one place which is located at a different place. Thus, it encourages stand-off annotation, that is, annotation that is expressed through mark-up that is defined at a different location.

Referential mark-up theoretically allows for an infinite number of independent layers of annotation. Stand-off annotation means that the mark-up and the text are not 'interleaved', that means mixed, but are physically kept at a different location. Usually, this leads to configurations where the several layers of annotation can be kept in separate files. Maintaining referential annotations is nonetheless also possible in single file formats as well, where annotation may be located in dedicated sections of the document. The *TEI P5: Guidelines for Electronic Text Encoding and Interchange* (cf. Burnard & Bauman (eds.) 2008: chapter 16) provide some good examples for mechanisms of in-document linking. These mechanisms can be exploited for stand-off annotations, but are also applicable for schemas with a multiple file approach where the annotation is kept in separate files (See *figure 23*). The only difference between in-document linking and separate files is

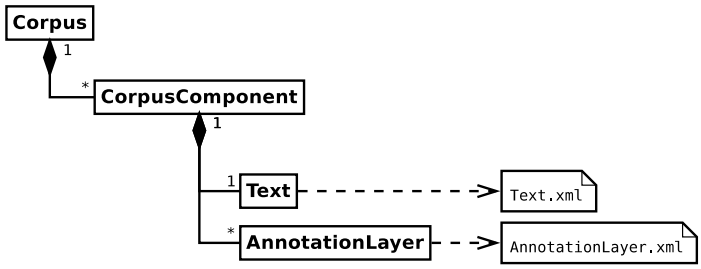



Figure 23: Typical file-layout for stand-off annotation. In this case it is possible to separate the annotation layers into multiple files.

that the reference has to be constructed to point outside of the file. Nevertheless, the mechanisms defined in XML co-standards like XInclude (cf. Marsh, Orchard & Veillard 2006) are sufficiently generic to reference resources at almost arbitrary physical locations. That means that in-document linking and linking to separate files can be handled by most current XML processors in a way that makes these two cases almost identical to the user. Nevertheless, a user of a resource has to make sure that all externally addressed files are available.

Referential mark-up does not have to rely on the physical order of the mark-up, but on references to a base-line (e.g. tokens in a text). Therefore, the hierarchy of structures and substructures does not have to be reflected in the physical layout of an annotated file. For this reason, it is common practise to arrange the mark-up in simple structures. As the structure of the mark-up, here, is itself not semantically important for the annotation, one of the simplest structures would be a list-like structure.

The structure of referential mark-up can be very simple and straightforward. Nevertheless, this appealing simplicity comes at a price. While the encoding of an arbitrary number of annotation layers is simple, the processing and querying of the annotation is more difficult. This starts with the fact that references have to be resolved to be understood. The most straightforward example for this would be the example of a human reader who

wants to directly read the corpus-files. Usually, as XML is based on human interpretable strings, this should not be an issue. But, as information with stand-off annotation might be distributed over several files, several files would have to be used in parallel to make use of the annotation. In this case it would require the readers to use at least two files and manage to resolve the references themselves. Therefore, reading and understanding two files synchronously is a cognitive challenge. This issue worsens with the number of annotation-layers used. On the one hand, referential mark-up opens the possibility to have an almost unlimited number of annotation-layers while, on the other hand, even few layers of annotation tend to be beyond adequate readability. Thus, the simplicity of referential mark-up might be appealing, but the information is very difficult to use. Stand-off annotation therefore usually has to be either merged, that means "internalised" (cf. Burnard & Bauman (eds.) 2008: section 16.9) into a single document or a custom user-interface designed to display the layers of annotation has to be developed. In any case, the use of multi-layered stand-off annotation generally requires significant effort to render it usable.

### 6.3.2 Hierarchical in situ Formats

Similar to the linear referential formats, the hierarchical format is suitable to express a wide range of annotations. The main feature of hierarchical formats is that the relation of the annotation to the text is expressed by the physical position of the used XML elements in the text. The annotation and the range of text, it is attributive of, is directly anchored in the text. Thus, the annotation can be seen as a graph anchored and related to the textual representation of the language. It can also, despite the fact that references are established by the concrete physical location of the mark-up and not by abstract referencing to the text, be regarded as an annotation graph as described in *formal framework for linguistic annotation* (cf. Bird & Liberman 2001).

Hierarchical formats are, in contrast to the referential formats, determined by their physical mixture of text and mark-up. This leads to a strong preference of single-file storage (See *figure 24*). If the annotated resource is divided in several parts, then, these parts delimit

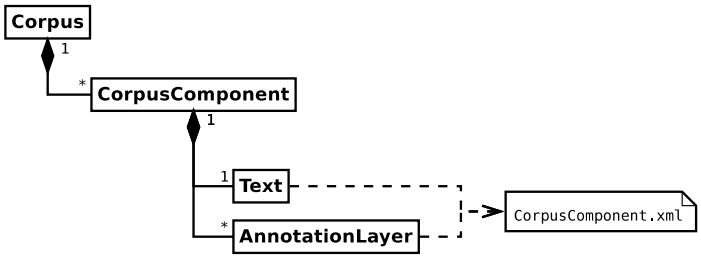



Figure 24: Typical file-layout for hierarchical annotation. The file-structure most conveniently corresponding to the data-structure is a set-up where all annotation is stored in a single file.

the boundaries between parts of the text and not text and annotation. A likely situation in which an approach like this would be applied, would be to reduce the size of individual files for the convenience of editing. A common delimitation in individual files would be by the definition of a chunking unit that is either determined by the data or by a sampling unit defined by the corpus design criteria. This strategy is found in almost all hierarchically annotated text corpora like VOICE, the BNC or MICASE. The alternative approach would be to have the complete corpus in a single file. This, nevertheless is not a sound option, as it leads to huge single file resources where it becomes more difficult to in- or exclude on the granularity level of the sampling unit. In the case of VOICE that would mean that it would be more difficult to contrast between domains by selectively choosing the files belonging to the individual domains for analysis. Furthermore, it can be guaranteed by the hierarchical structure of the mark-up that the resources may be merged back into a single file if required. Hence, splitting a resource into multiple files on the level of the sampling unit has in practice advantages but no disadvantages.

The possible levels of annotation in this approach are potentially limited. Unlike with referential formats where each annotation-layer does not necessarily add complexity to the format, hierarchical formats can get and in practice actually do get, much more complex with each added layer. The problem with this is, that the correspondence of start and end-tags in XML requires certain nesting requirements to be in place. In particular, this means the formal requirements of XML require that it has to be guaranteed that any element whose start-tag is after another element's start-tag but before its end-tag, has to be closed before the other elements end tag. In short, this means that elements have to be either fully comprised by each other or not to overlap at all. This has strong implications on layers of annotations that are in itself independent from each other. On the other hand it simplifies the processing model so that the structure is guaranteed to be a tree-structure where each element-node can be represented by a single leaf in the tree-shaped graph which itself can be regarded the annotation-graph of the file. The limitations in complexity of hierarchical mark-up have to be considered in the design for an annotated text resource.

Hierarchical mark-up has the advantage and disadvantage that the annotation and text is mixed in the same files. The disadvantage is that due to XML inherent nesting restrictions overlapping mark-up is as such not permitted. Thus, the level of concurrent annotations is per se limited. The limit, though, depends very much on the individual structure of the annotation and the density and likelihood of overlap between annotations. The greatest advantage is that for a human reader it is well possible to read the text and to see and understand multiple levels of annotation at the same time. In referential formats, in contrast, it would require the reader to work with several files and resolve references manually. The user-experience is therefore much closer to normal reading and the usual concept of reading a text. The only exception is that the reader has to process far more information encoded in special mark-up. The same positive aspect has to be expected for processing. Annotation

expressed in hierarchical mark-up can be processed in a meaningful way without the need to programmatically resolve any references. This results in two advantages. First of all, it allows the resource to be processed by tools that are neither aware of the particular semantics of referencing mechanisms nor any XML processing. Thus, tools working solely on the basis of the present textual information, like for example *WordSmith Tools* (Scott 2010), can be used without any preprocessing of the resource. Secondly, tools that are capable of interpreting XML-based mark-up, such as *XAIRA (XML aware indexing and retrieval architecture)* (Dodd 2010), can be much simpler in structure as they have to implement no methods to resolve references and can make use of the position of a node in a tree. This position of, for example, a given piece of text, can be used for simple tests whether some piece of text is in the scope of a particular mark-up. Thus, the greater complexity of the mark-up leads to more simplicity on the processing side. As long as the annotation's complexity is not beyond the possibility of being represented in hierarchical mark-up, the simplicity gained on the user's side is a strong advantage that makes hierarchical formats suitable and the better choice for encoding.

### 6.3.3 Relation Between Referential and Hierarchical Mark-up

Referential and hierarchical mark-up strategies can both be used to express annotations for language data. The difference is primarily the different approach, while both can have advantages and disadvantages. Many annotations can be expressed equally well in both formats. This means that an annotation that might be expressed equally well in either format establishes a relation between hierarchical and referential formats. Thus, in this case both formats can be mapped to each other. This relation is determined by following conditions: Stretches of annotations may nest, but otherwise not overlap. The transformation from hierarchical to referential mark-up is called 'externalisation' while the other direction is called 'internalisation'. "However: internalization is not clearly defined for all stand-off files, because the structure of the internal and external markup trees may overlap" (Burnard & Bauman (eds.) 2008: section 16.9.3). Nevertheless, internalisation is always clearly defined for stand-off annotation that has been derived from the externalisation of hierarchical single-document annotations. Therefore, the externalisation of stand-off annotation is, regarding the information, a loss-less operation whereas the internalisation is only loss-less if the nesting of elements is obeying above rules for the nesting of elements.

### 6.3.4 Implications for VOICE

VOICE features rich and dense annotations of language. Nevertheless, the advantages of single-file hierarchical mark-up for the texts outweighed the effort of having stand-off annotation. Nonetheless, VOICE features annotations that may span from locations within words and over complete words. If the word boundaries would be defined by elements spanning from the beginning to the word to its end, some of the linguistic annotation may

conflict. Especially annotation that may begin within a word and is valid for a longer stretch (i.e. a `speaking_mode`) would conflict. Therefore, hierarchical annotation is only feasible if it is possible to express the word-boundaries in a non-hierarchical fashion. For this purpose, VOICE adopted the strategy not to use XML-Elements but white-space[33] to delimit word-boundaries. But, as described above, annotation may at any point be externalised without any information loss at a later point. Therefore, VOICE decided to use hierarchical mark-up to express the annotation of the resource. In case the annotation has to be extended, specifically on word-level as it would potentially be the case with part-of-speech annotation, possible conflicts in the XML hierarchy have to be considered. A solution to any conflict on the XML level, would be that some parts of the annotation could be expressed referentially as stand-off annotation.

## 6.4 Selection of the Encoding Format

The selection of the encoding format is of fundamental importance for a language-encoding project. It determines the semantics of the information that may be represented in the resource, but also the way one can work with the data. Specifically, it has an effect on the way the structure of the encoding can be exploited for research purposes. This has implications in two areas. First of all the designers of a resource are required to fit the data as well as the annotation into the format. Secondly, the options the user has in investigating a resource are not only determined by the data and the user's research methods, but also by the way the data is represented and encoded in the corpus.

One important aspect that is relevant for the choice of encoding format is the great potential of resource saving when using and extending pre-existing experience in language encoding. This aspect is similar to the decisions for Brown, where, even though there had not yet been many predecessors in the encoding of a linguistic resource, the project has adapted and extended an encoding concept that was already in use for other purposes (see *6.1. History of text encoding formats p.56*). This way, a greater share of resources is available for questions specific to ones own research perspective and the specific goals of an encoding project. For VOICE this means, that by the extensive use of preexisting technology, a greater part of the available project resources could be dedicated to the specific needs of the ELF researchers of the VOICE project, but also the requirements of the wider ELF research community.

### 6.4.1 Requirements

The encoding format for VOICE would have to fulfil the following criteria. It should be

- capable of encoding most features of VOICE
- based on semantic categories, rather than for example formatting

[33] White-space characters are e.g. ordinary spaces or tabulator characters

- extensible, so that features that might not be readily available in the encoding, can be easily added[34]
- explicitly extensible, such that extensions to the format should be part of the design of the format. Therefore, extensions can be easily distinguished from the default encoding semantics
- after customisation capable of encoding all features of VOICE
- well documented, such that VOICE would not have to document every feature, but only the features where VOICE extends or changes the semantics of the encoding format
- widely used
- open, such that a user can extract parts of the data according to the specification without the need of reverse-engineering
- transparent, such that the format is usable with tools other than those that are specifically tailored for the particular encoding format
- free from vendor lock-in[35]. That means it should be possible to migrate to a different format at any time in the future to improve the sustainability

A format that is capable of encoding most features of VOICE could be found in projects with overlapping research interests. As there were no other specialised ELF corpora available, other corpus projects with similar requirements had to be evaluated. For this it was important to focus on some aspects that are specific to VOICE but not ELF. VOICE is a corpus

- of spoken English
- naturally occurring English captured in field recordings
- comprising complete speech events
- maintaining information on the speaker's language background and properties like age, gender and similar social factors
- maintaining information on the speech situation

Some projects, that had similar aims, albeit to a different extent in different areas, depending on the encoded features, are the British National Corpus (BNC) (cf. Oxford University Computing Services 2009), the American National Corpus (ANC) (cf. American National Corpus Project 2009) or the Michigan Corpus of Academic Spoken English

[34]The need for extensibility is not only important for the initial creation of VOICE, but also for extensions at a later time (see e.g. *Evaluating the applicability of existing POS practices for a corpus of English as a lingua franca (VOICE)* (Osimk forthc.) for a possible extension of VOICE).

[35]'Vendor lock-in' is generally understood to be the effect that data captured by one particular software is stored in a format particular to this sofware and the migration to a different sofware causes potentially high costs.

(MICASE) (cf. Michigan Corpus Linguistics 2010). Corpora that annotated the source-signal like AMI-Corpus (cf. AMI Project 2009) were disregarded, because the encoding approach differs fundamentally from the approach of VOICE as the AMI-Corpus relates the annotation to the recorded audio-signal as opposed to the transcribed language.

The BNC is one of the older corpora, therefore the corpus format has over the years evolved and undergone several transformations of the encoding format (cf. Burnard 2002). Today, the BNC as it is distributed on the sampler BNC Baby (cf. Burnard & Wynne (eds.) 2004) is encoded using a customisation of TEI P4. That means it is based on the first XML version of the TEI Guidelines. Before the shift to XML, the BNC was encoded in SGML-based TEI[36].

The ANC features, among many other text types, spoken language data. The ANC follows a stand-off annotation paradigm using the XCES format. This format is inspired by the TEI Guidelines, with many modifications and extensions. For the use of in-line mark-up in a single document, transformations are available that can merge several layers of annotation. Nevertheless, this only works for annotations that nest properly (See *6.3.1. Linear Referential Formats p.64*).

MICASE uses a format based on TEI P4 with modifications to the Document Type Declaration (DTD) (cf. Simpson, Lee, Leicher & Ädel (eds.) 2007: 10) to realise overlapping speech. The need for mechanisms to realise overlapping speech is similar to the needs of VOICE with the difference that MICASE data are not very interactive. Therefore, the requirements to mark-up expressing overlaps are more basic than for VOICE. Nevertheless, the fundamental handling of overlaps is very similar to VOICE and has inspired the representation in VOICE's corpus format.

All these Projects pointed more or less directly towards the use of a schema according to the TEI Guidelines. MICASE and the BNC with a direct application of TEI and ANC with the XCES format which is at least strongly inspired by the concepts behind the TEI Guidelines.

### 6.4.2 The TEI P5 Format

The Text Encoding Initiative (TEI) was founded in the late 1980s as a set of principles and was officially established and founded in 1988 with funding from the National Endowment for the Humanities, the European Commission, the Andrew Mellon Foundation and the Social Science and Humanities Research Council of Canada. (cf. Burnard & Bauman (eds.) 2008: section iv.2)

The first draft proposal (P1) was published in 1990. The second revised draft of the proposal (P2) in 1992 and the input and additions to these two proposals in 1994 as P3. The TEI Guidelines were developed within a large research community with

[36] Shifts in format are for XML corpora not problematic as it is possible to define a declarative equivalent transformation to a new format. See for example *P4 to P5 converter* (Rahtz 2007).

expert groups dedicated to specific areas of text-encoding. Therefore, the TEI Guidelines were a community driven collaborative effort that extensively researched on principles of text-encoding. Linguists as well as historians, conversation analysts, librarians, archive scientists and computer scientists contributed in their area of research with their particular encoding-needs in mind. The resulting comprehensive encoding principles were suitable to support a wide range of disciplines. Even though these principles were initially implemented in the recommended SGML based format of the P1 to P3 TEI Guidelines, they were not bound to the SGML technology but could be applied, as principles, to different mark-up languages. This became necessary at the end of the 1990s as

> in February of 1998 the World Wide Web Consortium issued a final Recommendation for the Extensible Markup Language, XML. Following the rapid take-up of this new standard metalanguage, it became evident that the TEI Guidelines (which had been published originally as an SGML application) needed to be re-expressed in this new formalism if they were to survive. (Burnard & Bauman (eds.) 2008: section iv.2 p5)

Even though this required very many adaptations, it proved the scientific foundation and the community support for the project mature enough to master a shift in technology. The result of this refactoring were the TEI P4 recommendations which transferred the concepts and thus the guidelines from SGML to the SGML based XML format. After this transition, the TEI started the work on P5, the current revision, which is the first version after the switch to XML which incorporates substantial new features. Especially, in regard to the transcription of speech, many concepts were substantially refined. For example, the enhanced linking and pointing mechanisms that are important to define and reference overlapping stretches of speech make the use for spoken corpora very attractive. As the development of VOICE and P5 happened synchronously it was attractive to adopt the recommendations even though, probably even because they were not yet finished. First of all, many features that are useful for spoken language were missing in P4 and planned to be included in P5. Secondly, the experience in the encoding of VOICE could be reported back to the developers and experts at the TEI-Council. Due to the complexity and the numerous options the TEI Guidelines provide for a wide range of encoding tasks, there were, as P5 was at the drafting stage when VOICE's encoding was settled, still some minor errors in the proposed schema. These minor issues were specific to the encoding of spoken language, which are for VOICE the most important parts of the TEI Guidelines. As the TEI user community actively working with spoken data appears to be small compared to the overall number of TEI users, these parts of the schema naturally receive less testing. From the questions raised in the official TEI-C public mailing-list (see Brown University 2011), it is obvious that the great part of encoding projects deal with manuscripts and written language.

It is important to note that the TEI P5 Guidelines provide the encoder with several alternative options for many of the aspects that have to be encoded. For example, there

are a number of different mechanisms in place to annotate overlapping speech. The VOICE project always tries to choose a generic approach which does not in itself impose a specific methodology on the user. The guidelines provided by the TEI provide a wide range of options that allows to choose the most suitable for the targeted resource.

### 6.4.3 Suitability for Encoding VOICE

An encoding based on the P5 version of the TEI Guidelines meets all requirements defined for VOICE (See *6.4.1. Requirements p.69*). Therefore, the VOICE team decided to model a customised encoding based on P5. The choices taken are described in the following and are defined in section *6.5. Mapping the VOICE Transcripts to VOICE XML (p.88)*. Other encodings, like XCES, would as well be a suitable choice for the encoding of VOICE, but TEI surpasses all other formats in regards of available documentation. This includes also individual feedback by active members of the TEI developer community.

### 6.4.4 Principles for Encoding VOICE

The following fundamental principles and strategies are used for the selection of components from the TEI inventory and the VOICE-specific extension of the TEI where necessary. Based on these principles, the encoding format of VOICE is assembled to meet the requirements of the annotation schema as it is expressed in VOICE transcripts (see *4. Redesigned Transcription Format p.29*).

- Each speech event is encoded in one self-contained `<TEI>` element in a distinct file (i.e. a TEI document)
- The complete descriptive information (meta-information) to the speech event is encoded in the same document within the element `<teiHeader>`
- Meta-information that describes an individual participating in VOICE is referenced with the pointing attribute `@sameAs` to a common description in an element `<teiHeader>` that describes the common features of the corpus
- Any addition of elements to the TEI-based encoding schema are kept in a VOICE specific namespace (See *6.4.6. Namespaces p.74*)
- All transcribed text is stored in XML text nodes in and between elements below the document's `<body>` element
- Alternative text, like for example a translation, is stored in XML element attributes
- Words are separated by white-space that is prepended or appended to text nodes containing the characters of the word.
- Text-nodes solely consisting of white-space do not separate words and can be used for formatting to render the XML files more readable for a human reader.
- Whenever possible, a feature or aspect should be encoded conforming to the TEI schema and not by introducing custom extensions.

Above principles are not only formulated to help with the creation and the encoding of VOICE, but also to make the resource more accessible for research. In many cases it is sufficient to remember above principles to know how to interpret the annotation. This facilitates the reliable work with the resource.

The basis of the VOICE text encoding is the structure of a minimal TEI file (See *figure 25*. All features that are selected from or added to the TEI encoding schema are fitted into

```
<TEI xmlns="http://www.tei-c.org/ns/1.0">
  <teiHeader>
    <fileDesc>
      <titleStmt>
        <title>title of the text</title>
      </titleStmt>
      <publicationStmt>
        <p>where and how a text is published</p>
      </publicationStmt>
      <sourceDesc>
        <p>description of the original data</p>
      </sourceDesc>
    </fileDesc>
  </teiHeader>
  <text>
    <body>
      <div>
        <u> i say yes </u>
      </div>
    </body>
  </text>
</TEI>
```

Figure 25: A minimal TEI P5 file

this basic structure.

### 6.4.5 Character Encoding

XML uses the UTF-8 character encoding as a default. The UTF-8 encoding is capable of encoding the whole UNICODE code-range (cf. Unicode Consortium 2010). Therefore, it is suitable for encoding a wide range of characters covering most known written languages. VOICE only uses characters from the basic Latin alphabet with the addition of the UNICODE ranges assigned to the encoding of the International Phonetic Alphabet (IPA) that are relevant for the encoding of onomatopoeia and pronunciation variation and coinages. Even though many other character encodings would suit the needs of VOICE, only the UTF-8 encoding is guaranteed to be compatible with any XML standard processor.

### 6.4.6 Namespaces

VOICE uses elements from different encoding schemes. Most of the Elements are defined in the mark-up vocabulary provided by the TEI, while a small set of elements are specific to the VOICE project. To make it easier for programs and hence users to distinguish the stock TEI elements from the VOICE specific elements, VOICE uses ‘namespaces’. In

XML every element has a namespace. A namespace is a string that is used to identify a set of elements belonging to a common encoding schema. Even though every string can be used as namespace when defining a schema, it is customary to use an Uniform Resource Identifier (URI). Namespaces are used to distinguish separate encoding schemas. Therefore they make it possible to use several schemas for the encoding of a single resource. They effectively prevent the risk of clashing tag names which would lead to ambiguous mark-up semantics. Namespaces can be bound to prefixes such that elements of different namespaces can be disambiguated (see *figure 26*). Namespaces are an important feature

```
<div xmlns="http://www.tei-c.org/ns/1.0"
     xmlns:tei="http://www.tei-c.org/ns/1.0"
     xmlns:html="http://www.w3.org/1999/xhtml">

  <!--
      - the default namespace is http://www.tei-c.org/ns/1.0
      - the prefix tei is bound to the namespace
      http://www.tei-c.org/ns/1.0
      - the prefix html is bound to the namespace
      http://www.w3.org/1999/xhtml
  -->
  <tei:p> this is a TEI encoded paragraph</tei:p>
  <html:p> this is an HTML paragraph</html:p>

  <p> this is also a TEI encoded paragraph </p>
  <p xmlns="http://www.w3.org/1999/xhtml"> this is also an HTML paragraph </p>

</div>
```

Figure 26: Example of namespaces in use

of all XML processing tools. VOICE uses two namespaces. The TEI namespace is the default for all elements defined in the TEI Guidelines and a separate VOICE namespace for elements and attributes additionally defined for VOICE.

- http://www.tei-c.org/ns/1.0 is the namespace for all elements that are defined in the TEI Guidelines
- Where the TEI namespace is not the default namespace, the prefix ‘tei’ is bound to the TEI namespace.
- http://www.univie.ac.at/voice/ns/1.0 is the namespace for VOICE’s additions and modifications to the TEI.
- The VOICE namespace is generally not defined to be the default namespace. The prefix ‘voice’ is bound to the VOICE namespace.

### 6.4.7 Use of Identifiers for Cross-referencing

VOICE uses an hierarchical document format. As hierarchical formats define the mark-up in situ (see *6.3.2. Hierarchical in situ Formats p.66*) matching the scope of the annotation, referencing, that is pointing to a different location, is not inherently necessary to define

the annotation of the text. Despite this, cross-referencing is used in VOICE for various purposes. First of all, it is used to make it possible to link information that mutually belongs to each other. For example cross-references align one temporally overlapping segment to the other segment it overlaps. Descriptive references to shared information are also common in VOICE. This is for example information as given in the description of a speaker within the text header. From the level of the utterance only a reference is given to the information shared by all utterances by this speaker. This is economic and improves the maintainability in case some of the information has to be revised for a later edition of a text, as it avoids the repetition of shared information. As a general rule, annotation on the text are not established with references in the mark-up. Nevertheless, descriptions and further information that more narrowly specify the particular instance, are linked via references. Information that may be referenced, that is pointed to from some other mark-up, is supplied with a unique value for the attribute `@xml:id`. The `@xml:id` attributes are used in VOICE for the identification of important elements. (See table 2 XML IDs in VOICE).

Table 2: XML IDs in VOICE

| Description | TEI Elements | Sample `@xml:id` from VOICE |
|---|---|---|
| texts | `<TEI>` | POwgd14 |
| notes | `<note>` | POwgd14_wc |
| text classification | `<category>` | POwgd14_ED |
| individuals | `<person>` | POwgd14_S1 |
| utterances | `<u>` | POwgd14_u_11 |
| overlaps | `<seg>` | POwgd14_ol_419 |
| shifts in voice quality | `<shift>` | POwgd14_s_48 |
| indications of other-continuation | `<anchor>` | POwgd14_oc_5 |

The `@xml:id` attributes are created according to the following production rule in EBNF:

| Component | | Definition |
|---|---|---|
| XMLID | := | DomainID, SpetID, UniqueTextNumber, (SubType, Numbering?)? |
| DomainID | := | alpha_uchar[37], alpha_lchar |
| SpetID | := | alpha_lchar, alpha_lchar |
| UniqueTextNumber | := | num_char+ |
| SubType | := | "_", alpha_lchar+ |

[37] See table 1 for the definition of `alpha_uchar`, `alpha_lchar`, `alpha_num` and `num_char`

| | | |
|---|---|---|
| Numbering[38] | := | "_", num_char+ |

The production rules for the values of `@xml:id` in VOICE reveal information on the kind of data that is referred to or identified. This is not a requirement by the selected encoding technologies, such as XML or the TEI Guidelines, but a deliberate choice for the encoding of VOICE. This is especially useful when working with analysis software that in many cases returns parts of the documents that match a particular query. As some aspects of the data is encoded in the `@xml:id` and is in VOICE granted to be constructed according to above production rules, it is possible to derive important aspects from this identifier. Several features of a particular instance can be derived without having to resolve the reference to the referenced object. In particular, it is granted that an `@xml:id` identifies the text, the domain, the speech event type and reveals the kind of information that is referenced by the `SubType`.

### 6.4.8 Language Codes

The core information that is provided for the individual speakers in VOICE includes the first languages of the speakers. As information on languages, as for example the first languages of the speakers, is central to language and the encoding of language, VOICE uses language codes to encode languages in the meta-data. In this, the VOICE corpus format adheres strictly to the language codes recommended by the TEI. Generally, every speaker in VOICE has at least one language code for the known languages. The speaker in *figure 27*, for example, is a bilingual speaker. The `@tag` attribute of the element `<langKnown>`

```
1 <langKnowledge xmlns="http://www.tei-c.org/ns/1.0">
2   <langKnown level="L1" tag="spa-ES"/>
3   <langKnown level="L1" tag="cat-ES"/>
4 </langKnowledge>
```

Figure 27: Example of the encoding of languages in VOICE. This speaker is a bilingual speaker of Spanish and Catalan as it is spoken in Spain.

indicates the particular language, whereas the `@level` attribute indicates that the language is a L1 for the particular speaker. This information is present for the sake of explicitness, as in VOICE only information on the first languages of speakers is encoded. That is, no information on languages acquired later as second languages may be found in the corpus. That means, even though the speakers use English as a lingua franca, their knowledge of this and other second languages is not explicitly indicated in the corpus header.

The values, recorded in the language tag, are defined as recommended in *BCP47: Tags for Identifying Languages* (Phillips 2009). This comprehensive recommendation includes methods defining and specifying language codes ranging from codes for extinct languages

[38]The numbers are selected in ascending order beginning with 1 in the order of the document for each numbered `SubType` in a particular text.

over currently used languages and variants to artificially constructed languages. The language tags specified by this recommendation are generated using construction rules. According to the specification the 'tags' are constructed on the basis of a combination of 'subtags'. In VOICE, only the encoding methods for the definition of the primary language subtags and an optional second subtag indicating a region are used. A language subtag, following this schema would for example be 'spa' for Spanish followed by an optional subtag indicating a region, as for example 'ES' for Spain. Even though the regional subtag of the speaker can indicate possible linguistic differences, "[...] they do not imply that a significant dialectical boundary exists [...]" (Phillips 2009: 62). Even more emphasis has to be put on this for VOICE as the country code given for the regional information in VOICE is usually derived from the provenance of the speaker. I noticed that this could be argued to be a legitimate use of the subtag, but this definition could still lead to misunderstandings and untenable assumptions on the language background. Therefore, any claims based on the language, especially when related to the regional background of speakers have to be checked carefully. BCP47 offers many other optional subtags for the encoding of various aspects of a particular language. Nevertheless, they are beyond the scope of what VOICE aims to cover and therefore, consequently, not used.

### 6.4.9 Encoding of Times

Even though the annotation of VOICE is not temporally aligned, some times are annotated in the corpus format. Temporal information is either of the type 'duration' or 'timestamp', when relating to a point in time. Durations generally follow the recommendation to prefer the standards referenced in the TEI Guidelines (Burnard & Bauman (eds.) 2008: section 3.2.6). These standards, similar to the standard for language codes, allows for duration formats for different uses. Some uses, like measuring in years and months, are not applicable in the context of VOICE. Therefore, they are not explained here, even though they are allowed in TEI based encodings. Two variants are used for durations in VOICE: One encoding for longer durations measured in hours, minutes and seconds and a shorter encoding for durations measured in seconds.

Table 4: Durations in VOICE

| Used for Element | Format | Description |
|---|---|---|
| `<incident>` | PT3S | A duration of three seconds |
| `<pause>` | | |
| `<recording>` | PT01H22M41S | A duration of one hour, twenty-two minutes and forty-one seconds |

Timestamps indicating a given time are relative to the recording medium and not normative. That is, timestamps in VOICE are only additional information that can be used

to remember temporal locations more easily or get a rough impression of the duration of the transcribed portions. They were used during transcription and are only retained in the final corpus format for convenience. Timestamps are only used on the `<div>` element, which corresponds in VOICE to the boundaries of a recorded medium. These times are used to indicate from which point relative to the beginning of the recording to which later point the transcript ranges. This is useful as the speech event might begin some time after the recording started. Furthermore, when parallel recordings exist, the times indicate when and for how long a specific medium was used as a source for transcription. The temporal description of `<div>` is given in its attributes `@voice:end` and `@voice:start` in the VOICE namespace. This follows a simple minutes and seconds based format. Minutes and seconds are separated by a colon.

Table 5: Timestamps in VOICE

| Used for Element | Format | Description |
|---|---|---|
| `<div>@voice:start`<br>`<div>@voice:end` | 01:30 | The time one minute and thirty seconds after the beginning of the medium |

### 6.4.10 Description of Participants

One of the assets of VOICE is that the assignment of individual participants is very transparent. That means that it is possible to locate individuals and identify their occurrences in other speech events. As in VOICE there are groups of events that are recorded either in parallel or sequence, some individuals can be present as speakers in several speech events. That means, it is possible to select a sample based on individual speakers. Especially when working with rather small samples, a single speaker that is in some aspect exceptional can bias the work. In these cases it is very helpful to be in a position to single out which contribution is coming from a specific speaker.

The identification of speakers for the better understanding of a possible bias in the data is just one aspect. The second and more important aspect is that a set of additional information is available for each speaker in VOICE. That is, the participant description consists of information on:

- groups of speakers
- identified speakers
  - age
  - sex[39]

[39]The term ‘sex’ is used in accordance with the TEI Guidelines. Generally, this relates to concepts that would more fittingly be described as ‘gender’.

  - occupation/profession
  - role in speech-event
  - language background
- non-identified speakers
  - if it is likely that this is one of the identified speakers, a reference to this speaker is given
  - if possible, the sex of the speaker is given
  - generic groups of speakers contributing the same at the same time are defined (i.e. 'SS')

The range of information that is available on the speaker level is illustrated in *figure 28*. The element <person> is a container for any information on the speaker level. First

```
<person role="participant" sameAs="#P94" xml:id="PRpan13_S17"
        xml:ns="http://www.tei-c.org/ns/1.0">
  <age value="3">35-49</age>
  <sex value="2">female</sex>
  <occupation>professor of economics</occupation>
  <langKnowledge>
    <langKnown level="L1" tag="por-PT"/>
  </langKnowledge>
</person>
```

Figure 28: Example of the encoding of an identified speaker in VOICE. This speaker is a speaker of Portuguese as spoken in Portugal.

of all, there is a set of attributes to this element that indicate the speaker's @role and a unique identifier @xml:id that is used to reference this speaker from her contributions. The attribute @sameAs indicates the entry in the corpus header related to this speaker. Apart from the specification of the role, these entries are the same and are used to establish a link between all speakers in different texts that are in fact the same individual. That means that it is possible to unambiguously identify an individual as a speaker in different text. All of these related speakers (i.e. the same person) have the same value for the attribute @sameAs in the individual texts. Furthermore, additional elements within <person> specify further details on the speaker. For VOICE, this is generally the

- <age> element, relating the speaker to an age-group
- <sex> element, specifying the gender of the speaker
- <occupation> element, specifying the occupation or profession at the time of recording. This information is not to be confused with the role in the speech-event, which is given in the attribute @role of the element <speaker>
- <langKnowledge> element, specifying the known language-background of a speaker. That is, her first languages

### 6.4.11 Representation of Stratification

VOICE is a stratified corpus, that is, it has an internal balanced structure. The main criterion for this stratification is the association of the corpus texts with a domain. That is, VOICE is divided in partitions along these *domains*. These are defined according to the sampling criteria as briefly described in the corpus-header in the element `<samplingDecl>`. There, the balance is described as follows:

> As to the sampling method used, subgroups of the target population were identified on the level of domain and target proportions specified for these as follows: Educational [(ED)] 25%, Leisure [(LE)] 10%, Professional-business 20% [(PB)], Professional-organizational [(PO)] 35%, Professional-research/science [(PR)] 10%. (VOICE Project 2009: corpusHeader)

In the corpus, these domains are referenced by a pointer to the defined domains in the corpus header. The reference can be intuitively understood, as the abbreviations for the domains and speech-event types are used to establish this reference (see *figure 29*).

```
1 <textClass xml:ns="http://www.tei-c.org/ns/1.0">
2   <catRef target="#EDcon4_ED␣#EDcon4_con"/>
3   <catRef target="#ED␣#con"/>
4 </textClass>
```

Figure 29: Example of the encoding of the stratification of a speech-event. The two XML elements `<catRef>` link to the description of the available domains. The first links to the description within the text header and the second to the equivalent within corpus header.

### 6.4.12 TEI Components

The TEI is a customisable framework for language encoding. It provides several distinct mechanisms of specifying which elements specified by the guidelines may be available in a specific encoding project. In most cases only a limited subset of the TEI is required for the encoding. Therefore, it is good practice to enforce this restriction by defining an appropriate schema with only the required elements. This prevents the unintended use of available elements and helps to prevent a diverging encoding practice within a project.

Elements within the TEI infrastructure can be selected at several distinct levels. Available elements are grouped into modules with elements that are required for similar encoding tasks. Available modules are among others: ‘core’, ‘spoken’, ‘drama’ or ‘analysis’.

#### 6.4.12.1 Components Selected

The TEI has historically a stronger focus on the encoding of written material, especially the digitisation of printed texts. For this reason, components for the encoding of spoken language are in some parts spread over modules that are related to its printed representation. Therefore, the selection of modules for a spoken language corpus such as VOICE mirrors

the fact that the TEI has its origin in the encoding of written language. Consequently, most elements required for the annotation of VOICE are present, but distributed over several modules. As a TEI customisation allows for the inclusion and exclusion of particular modules and elements, the VOICE customisation is essentially a selection of applicable modules with a subsequent exclusion of the elements in the selected modules that are not required.

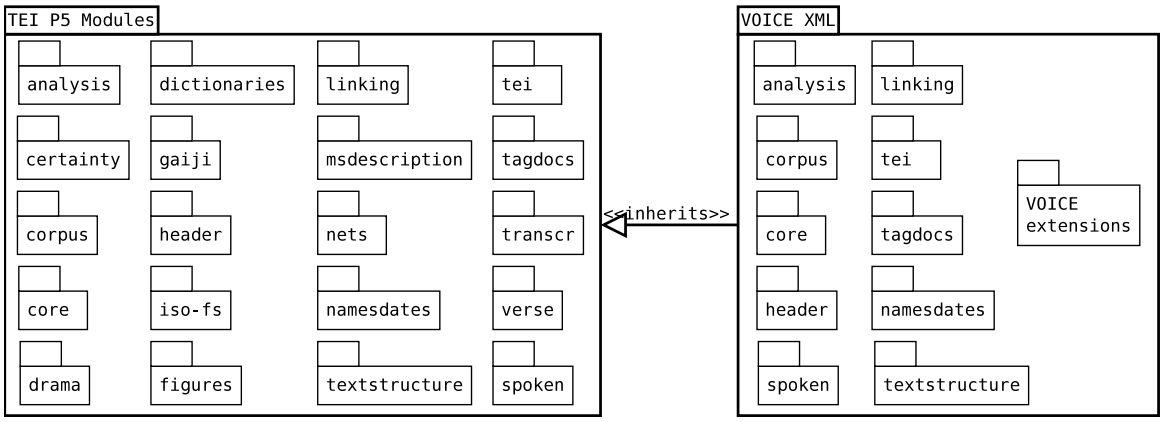



Figure 30: Components available in the TEI and modules the VOICE XML schema builds upon.

The elements required for the encoding of VOICE are distributed over many of the available TEI modules, where from some modules only few elements are actually used in VOICE. To avoid encoding errors, especially for later manual editorial work, elements and attributes that are not used for VOICE but defined by the TEI are removed from the VOICE XML schema. See table 6 List of TEI modules[40] for a list of available modules and a reference to the respective chapters of the TEI documentation for these modules.

Table 6: List of TEI modules

| Name | Description | Chapter in TEI Guidelines |
|---|---|---|
| *analysis*[41] | Analysis and Interpretation | 17 Simple Analytic Mechanisms |
| certainty | Certainty and Uncertainty | 21 Certainty, Precision, and Responsibility |
| *core* | Common Core | 3 Elements Available in All TEI Documents |
| *corpus* | Metadata for Language Corpora | 15 Language Corpora |
| dictionaries | Print Dictionaries | 9 Dictionaries |
| drama | Performance Texts | 7 Performance Texts |
| figures | Tables, Formulae, Figures | 14 Tables, Formulæ, and Graphics |

[40] This table is taken verbatim from *TEI P5: Guidelines for Electronic Text Encoding and Interchange* (Burnard & Bauman (eds.) 2008)

[41] *Emphasis* is used for modules relevant for VOICE

List of TEI modules(cont.)

| | | |
|---|---|---|
| gaiji | Character and Glyph Documentation | 5 Representation of Non-standard Characters and Glyphs |
| *header* | Common Metadata | 2 The TEI Header |
| iso-fs | Feature Structures | 18 Feature Structures |
| *linking* | Linking, Segmentation, and Alignment | 16 Linking, Segmentation, and Alignment |
| msdescription | Manuscript Description | 10 Manuscript Description |
| *namesdates* | Names, Dates, People, and Places | 13 Names, Dates, People, and Places |
| nets | Graphs, Networks, and Trees | 19 Graphs, Networks, and Trees |
| *spoken* | Transcribed Speech | 8 Transcriptions of Speech |
| *tagdocs* | Documentation Elements | 22 Documentation Elements |
| *tei* | TEI Infrastructure | 1 The TEI Infrastructure |
| textcrit | Text Criticism | 12 Critical Apparatus |
| *textstructure* | Default Text Structure | 4 Default Text Structure |
| transcr | Transcription of Primary Sources | 11 Representation of Primary Sources |
| verse | Verse | 6 Verse |

#### 6.4.12.2 Selected Elements

The following tables list the most important Elements and their purpose for the encoding of VOICE. These tables are not comprehensive, but reduced to the essential structuring elements to make the general structure apparent. As these Elements are directly taken from the *Guidelines for Electronic Text Encoding and Interchange* (TEI-Consortium 2007), a more general description can be found there. Nonetheless, the brief description next to each of the elements should help to understand the most important components and their basic meaning.

Table 7: TEI Elements establishing the general structure.

| Element | Use |
|---|---|
| `<teiCorpus>` | This element is used to establish the corpus framework for VOICE. That is, all descriptive information concerning the overall structure of the corpus is stored within this element within a toplevel `<teiHeader>` element. The individual components of the corpus, that is the individual transcribed speech events in VOICE, are kept in seperate files and only referenced from here. |

TEI Elements establishing the general structure.(cont.)

| | |
|---|---|
| <TEI> | The main encoding unit for the individual speech events. All information relevant to a particular speech event, meta-data as well as transcribed and annotated speech are located within this element. |
| <teiHeader> | The meta-data for the event is stored in a location separate from the transcribed data. All meta-data, descriptions of the event are located within this header. |
| <body> | The transcribed and annotated language data is all grouped below this element. |
| <gap> | A gap between transcribed portions of speech. Reasons for gaps are missing parts in the recording, poor audio quality or longer portions of monologue that are excluded from transcription. The duration of the gap and a description is given in the attributes of the element. |

Table 8: Essential sections of the header

| Element | Use |
|---|---|
| <fileDesc> | Descriptive information on the file. |
| <titleStmt> | Information that relates to the work's title and its editors, funders and sponsors. |
| <respStmt> | Describes a responsibility of a person in the creation of VOICE. |
| <editionStmt> | Information on the edition of the resource. |
| <publicationStmt> | Everything relating to the publication and availability of VOICE. |
| <notesStmt> | Descriptive notes on the speech event. Among others, a short description of the context of the event is to be found for every event. |
| <sourceDesc> | Information on the used recording equipment and the date of the recording. |
| <encodingDesc> | Defines the framework of text classes representing the stratification of VOICE and lists the tags used for the annotation of a particular text. |

Essential sections of the header(cont.)

| | |
|---|---|
| <profileDesc> | Establishes a profile of the speech event. It places the event into the stratification framework of VOICE, that is, it assigns a domain and a speech event type. Furthermore a description of the setting, the locale and all participating persons, including their language background, is part of this descriptor. |
| <revisionDesc> | Information on changes, steps and responsibilities in the editing of VOICE. |

Table 9: Elements used for the encoding of language

| Element | Use |
|---|---|
| <u> | The texts are split into utterance segments. Each utterance is assigned to a specific speaker. |
| <c> | Used in VOICE to mark intonation and lengthening. |
| <pause> | Indicates a pause in speech at the place of occurrence. |
| <emph> | Indicates strong emphasis, as transcribed with uppercase letters in the VOICE transcripts. |
| <shift> | Shift in voice quality or speaking mode. The different types of shift are indicated by attributes to this element. |
| <seg> | Indicates a segment of speech. These segments are used in combination with pointing attributes to align concurrent contributions by different speakers. |
| <supplied> | Supplied by the editor. In VOICE this is used for anonymisation and the insertion of replacement character “x” for unintelligible speech. |
| <unclear> | Speech tagged as unclear was particularly difficult to understand by the transcriber, but there was enough confidence not to use <supplied>. |
| <vocal> | Used to indicate vocalised, semi-vocalised and other speaker noises. |

Elements used for the encoding of language(cont.)

| | |
|---|---|
| <w> | In the TEI this is the generic mark-up for word-like segments. As VOICE uses complex mark-up, the words are separated by white-space. Nevertheless in some cases, as with spelt words, a different mark-up was required. The <w> element is used for spelt words in VOICE. It influences white-space processing, such that the white-space used in w delimits the boundaries between the spelt characters or syllables. |
| <anchor> | Anchors are set in VOICE at the beginning or the end of utterances to connect them when the transition between the utterances would be latching. This is a deviation from the TEI, as the @transition attribute does not allow to specify which two <u> elements are latching. Thus the default strategy could not be used as in cases with several simultaneous speakers, it would have been ambiguous. VOICE uses pointers from and to anchors to solve this issue. |
| <incident> | A contextual event. The textual description of the event is given in the VOICE specific @voice:desc attribute. |
| <foreign> | A stretch of language that is recognised to be in a language other than English. The attributes indicate the language, whether it is the L1 of the speaker and give a translation to English if possible. |

#### 6.4.12.3 Deselected Elements

The TEI modules that are selected to provide all necessary elements for the encoding of spoken languages, contain far more elements than needed for the encoding of spoken ELF. The generic analytic module serves as an example. There, many elements are defined that are not used in the VOICE encoding. For example, elements for phrases and sentences are defined which are not used in the VOICE annotations and therefore excluded from the TEI Customisation of VOICE. As an exhaustive list of deselected elements would not be helpful in the documentation of what the encoding scheme is, as opposed to what it is not, readers should refer to the appendix *VOICE TEI Customisation (p.150)* for details on the deselected elements.

#### 6.4.12.4 Extensions specific to VOICE

Only few features of VOICE had no satisfactory equivalent in the TEI Guidelines. Elements and attributes were added to include these features in the encoding of VOICE. Some of

the elements are added as a distinct feature, like the pronunciation variation and coinages (PVC, see *4.2.21. pvc p.40*), for which there is no unique equivalent. Other parts were added because they contradicted the encoding principles (see *6.4.4. Principles for Encoding VOICE p.73*). This is for example the description of contextual events (i.e. <incident> in the TEI Guidelines). In this case the description would be plain element content and not put in an attribute. Therefore, the description would interfere with the text when used with XML agnostic software like *WordSmith Tools* (Scott 2010). When it is put in an attribute, as it is implemented for VOICE, the attribute content is hidden for language processing tools that are designed to use just the transcribed language. For this reason all descriptive elements were moved into a custom attribute @voice:desc that is equivalent to the <desc> element in the TEI Guidelines.

The following table shows all deviations from the TEI recommendations. Where the use of an attributes (starting with an "@"-character) is described, the corresponding elements are put in the table row above the attributes.

Table 10: List of Extensions for VOICE

| Element | Attributes |
|---|---|
| <voice:track> | Marks a track of the original recording medium. The attributes @type, @medium and @num provide information on the used medium type (e.g. md for MiniDisc), the medium number in a sequence of media and the number of the track in a sequence of tracks of a medium. |
| <voice:to> | Marks some portion of speech that is directed explicitly towards another identified speaker in the speech event. The attribute @who is to be used for a pointer to the description of the person in the text header the speech is directed to. |
|  | |
| @voice:medium_type | The type of the media that are used for the transcribed part |
| @voice:start_medium_number | Number of the first medium of the transcribed part |
| @voice:start_track | Number of the first track of the medium of the transcribed part |
| @voice:start | Time of the beginning relative to the first track |

List of Extensions for VOICE(cont.)

| | |
|---|---|
| @voice:end_medium_number | Number of the last medium of the transcribed part |
| @voice:end_track | Number of the last track of the last medium of the transcribed part |
| @voice:end | Time of the end relative to the last track |
| <vocal> | |
| <incident> | |
| <gap> | |
| @voice:desc | A prose description of the previous elements. This is equivalent to the <desc> element as defined in the TEI Guidelines |
| <vocal> | |
| @voice:syl | The duration in an approximation in syllables. This is most frequently used for the encoding of laughter. |
| <w> | |
| @voice:mode | Descriptive of the speaking mode a word is uttered in. This is only used with the value 'spelt' in VOICE |
| <supplied> | |
| @voice:ipa | A representation of the uttered stretch, especially in case of unintelligibility, in IPA characters. |
| <foreign> | |
| @voice:translation | Whenever possible foreign speech has been translated to English. |
| @voice:ipa | A representation of the part in a foreign language, especially in case of unintelligibility, in IPA characters. |

## 6.5 Mapping the VOICE Transcripts to VOICE XML

We have so far discussed the transcription format and the corpus format. While the most difficult part in the process, as it is the most analytic and time-consuming, is the initial transcription and proofing of the transcripts, this is not were the process stops. This is due to the decision that the transcription format is not the same as the final format of the product. Therefore the transcription has to be taken as input to a transformation process which yields the corpus texts as an output. This transformation process can be broken down into distinct

steps. The steps immediately following the transcription is the formal validation of the transcribed texts (see *4.1. Formalisation of VOICE Transcripts p.32*), parsing the syntax tree of the input transcripts and transforming it to an XML output tree. This section is concerned with the specifics of the transformation process.

### 6.5.1 Relations Between Source and Target Objects

Every annotation that can be expressed with the *Mark-up Conventions* has a corresponding representation in the target VOICE XML corpus format. For example, the annotation `pause` is rendered as '(.)' in the transcription conventions and '`<pause/>`' in the corpus format. Furthermore, any structure of interrelated annotation, such as annotation that is part of some other annotation, has a corresponding structure in the corpus format. That means that dependencies of annotation that are required in one format map onto dependencies in the other. For example the annotation `pause` can only occur as being a part of the annotation `utterance`.

Any annotation in the transcription format has an equivalent that is structurally similar in the corpus format. That means, even though the mark-up is different, it is organised according to the same principles. This holds true even though the vocabulary of the two mark-up schemes differ. Thus, the structural similarities can be transformed relatively directly into the corpus format. Despite the fact that the vocabulary of both mark-up schemes differ, most parts can be directly mapped onto the corresponding structures in the corpus format without changes to their order (see *figure 31*).

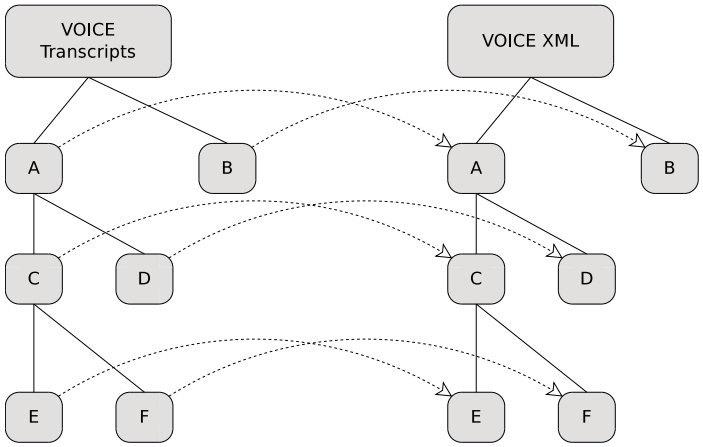



Figure 31: The default correspondence between VOICE transcripts and the XML representation is a direct relation. Every object in the source format relates to an equivalent object in the target format while the order of the objects in both formats is identical.

Some annotations in VOICE transcripts have no direct equivalent in the TEI-based corpus format. That is, there is no direct one-to-one mapping available for these annotation types. Nevertheless, these types have equivalent forms in the corpus format, where two annotation types can be used in conjunction to model the source annotation (see *figure 32*). The

most frequent type of annotation that is mapped in this fashion are annotations of the type `speaking_mode`. When mapping VOICE transcripts to an equivalent representation a

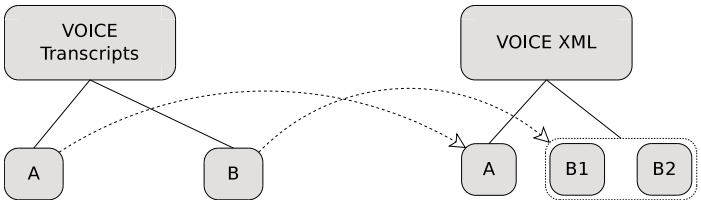



Figure 32: In some cases, as for example speaking-mode, it is necessary to model the annotation of VOICE transcripts with two distinct and independent annotations in VOICE XML. The figure shows that the annotation $B$ is mapped to $B_1 + B_2$ in the transcript. The annotation `speaking_mode` would not be mapped to a single XML element but to two independent `<shift>` elements.

complete parse of the annotation has to be performed. That means that every component of a transcript is analysed according to its position within the transcript. Based on this analysis, it is possible to validate the syntax of the transcripts as described in the following.

### 6.5.2 Validation

As VOICE transcripts follow a rigid and formally defined format (see *4.2. Components of VOICE Transcripts p.33*), it is possible to validate the structure of a particular transcript. 'Validation' in this context means that it is possible to systematically and programmatically test whether the transcript follows the defined formal structure. For this systematic validation, it is necessary to have an implementation of the specification. That is, a system that is capable of detecting and ensuring the formally defined rules. In VOICE, as the transcription format and corpus format is not the same, this function is performed by the VOICE Transcript Parser (see *VOICE Parser p.106*). It provides the reference implementation of the mark-up language specification that is used for the VOICE *Mark-up Conventions* (VOICE Project 2007) and a means for the transformation to the VOICE corpus format. Hence, as it is based on the expected formal structure of the transcripts, it can be used for the formal validation of the transcripts. In practice this means that wherever the parser can not unambiguously read the source transcripts, this is an indication of an error. In many cases these errors are related to minor formal flaws in the transcripts that lead to slight ambiguities. Usually these ambiguities are hardly noticeable for a human proof-reader, but are problematic in computational environments such as text corpora or corpus-analysis tools. In rare cases though, the reported errors show that some aspect that is required according to the transcription conventions has not yet been correctly implemented. That means, a failure of the VOICE Transcript Parser to read the transcripts usually leads to a correction of the transcript. But, as complex systems tend to contain minor errors, the behaviour of the program has not been taken to be a supreme authority. In case of

doubt, the program itself had to be evaluated and in some cases corrected to match the definition of the mark-up. The processing of the transcripts with the VOICE Transcript Parser does not generally include a qualitative evaluation of the transcripts. This reveals two important aspects of formal validation. First, in most cases human intervention is necessary to resolve an error in transcription. Secondly and most importantly, formal validation does not guarantee that a transcript is necessarily – judged by qualitative criteria – sound. Nevertheless, for the process of proof-reading the transcripts, errors reported during the formal validation phase provided many indicators to parts where further corrections of the source texts were found to be necessary.

The qualitative evaluation, and hence a validation of the qualitative information contained in the transcript is to a great degree beyond the scope of the formal validation that is possible with a computer algorithm. It is nonetheless evident that the formal specification of the transcription format follows the rationale of the qualitative descriptive needs. The reason for this is that the format is modelled after and determined by these needs. Consequently, it is not astonishing that formal mistakes can indicate issues relating to qualitative descriptors. For this reason, we have noted above that qualitative evaluation is not per se impossible, but is only possible at a general level.

In the area of qualitative evaluation, the VOICE Transcript Parser implements some consistency checks and heuristics to spot possible issues. One the one hand, some of the heuristics lead to hard program termination (i.e. hard failures), that is errors that have to be corrected to proceed processing. On the other hand these heuristics trigger warnings relating to possible mistakes, that is structures that might be formally possible but are known to be problematic or very difficult to transcribe. For this reason most of these consistency checks are related to structures that are most likely to occur in situations that are difficult to represent in written form (e.g. annotation of the type `overlap`). The following list is a selection of typical examples when the VOICE Parser would print a warning message or raise a run-time error.

- An `overlap` does not correspond to another `overlap` by a different speaker
- An overlap corresponds to more than two `overlap` annotations by other speakers
- The numbering of `overlap` has been reset (see *4.2.28. overlap p.41*)
- A `speaking_mode` has been used that is not in the closed list of standard speaking modes (see *4.2.36. speaking_mode p.42*)

The first item in the above list is clearly an encoding error. Therefore, the VOICE Parser would terminate the transformation and thus force the user to analyse the problem and change the transcript accordingly. This condition would be called an ‘error’ during the transformation. The second item has a very different quality. Here, the VOICE Parser warns the user of a very unlikely, albeit possible situation. The VOICE Parser thus continues the parse-run but emits a warning message. This would be called a ‘warning’ during

the transformation. Consequently, the user is advised to check whether the warning can be safely ignored or indicates a problem with the encoding. The third and fourth item are informative of the current progress of the VOICE Parser and are in most cases not problematic.

The validation with the VOICE parser was an important step after the second proofing phase that had been performed by experts, that is to say, by human agents. Similar to the *Mark-up Conventions* and the rigid process of transcribing and two phases of in-depth proof-reading the VOICE Parser is one component that helps to increase the reliability of the produced linguistic annotations.

### 6.5.3 Using the Parse-tree for the Mapping to VOICE XML

In the preceding section we have seen that the VOICE Parser can be used, as a secondary effect, for the validation of the mark-up. Nevertheless, its primary purpose is the transformation from the source format (i.e. VOICE transcripts) to a target corpus format (i.e. VOICE XML). Hence, this section is dedicated to the process that is used for this transformation.

We have also seen that the general mapping of source and target can in most cases be established with a trivial one-to-one relation. Only in few cases, some of the annotations need to be split and slightly rearranged (see *6.5.1. Relations Between Source and Target Objects p.89*). This generally very simple relationship has a direct influence on the implementation of the transformation as it simplifies the process. Generally speaking, some of the steps that are conceptually distinct can under these circumstances be performed in one pass, which greatly reduces the complexity. For a better understanding of this, it is necessary to examine the steps that are in principle necessary to transform an input format into an output format. First of all a parser component is required that reads the input and creates a parse tree according to the specification of the input format. As a second step, it might be necessary to control dependencies of input items that are not explicitly defined by the input format, but by restrictions on this input format. The third phase is a transformation of elements from the source parse tree to an output tree. The last step of this process is to serialise this output tree, that is, to create a sequence of output characters from it. Usually, this in effect means that the content is either printed or written to a file. But, as input and output format stands in a direct relation, some of the steps can be merged (see *figure 33*). That means that it is not necessary to create an intermediate representation of the input parse-tree, but the output can be produced directly during the input phase. In this configuration, for every definition of the input format the optional constraints are checked and the transformation to the output-tree is performed immediately. That is, whenever a known structure from the input format is recognised, the corresponding object for the output-structure is instantiated. The VOICE Parser (see appendix *VOICE Parser p.106*)

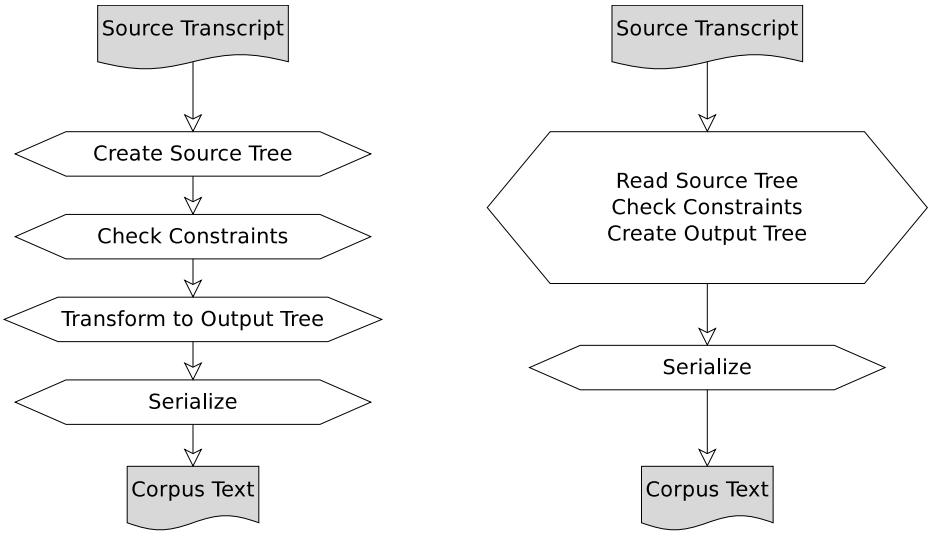



Figure 33: General and simplified processes for transforming a defined input format to a defined output format. In the simplified form no explicit representation of the input tree is built, but the output is generated directly when an input component has been successfully read.

interleaves the steps of parsing the input, checking optional constraints and creating the output-tree.

### 6.5.4 Inclusion of Related Data From Non-transcript Based Data Sources

For now, this chapter has only focused on transcribed data. VOICE as a corpus-resource comprises a range of further information, as we have seen in *5. The VOICE Meta-data Database (p.45)*. This data can be very useful when working with the corpus and is in many cases necessary to strengthen evidence-based inferences. The meta-data provides additional information that can influence the sampling and selection of data for a particular study as well as the interpretation of the texts in qualitative research contexts. In this part we will see how the meta-data from the database is extracted and matched to the relevant speech events in the corpus.

After the VOICE Parser has transformed a transcript to its equivalent in the targeted corpus format, a great part of the data that is available complementary to the transcribed speech-event is not yet available in the corpus text. As this data is not immediately present within the transcript, it is necessary to retrieve this data from its source and fit it at the appropriate places into the corpus format. That is, to merge both data-sources, so that the data is in place according to the specification of the corpus format. The data that is included from an external source is stored in a meta-data database (see *5. The VOICE Meta-data Database p.45*). That means, the VOICE Parser requests the information related

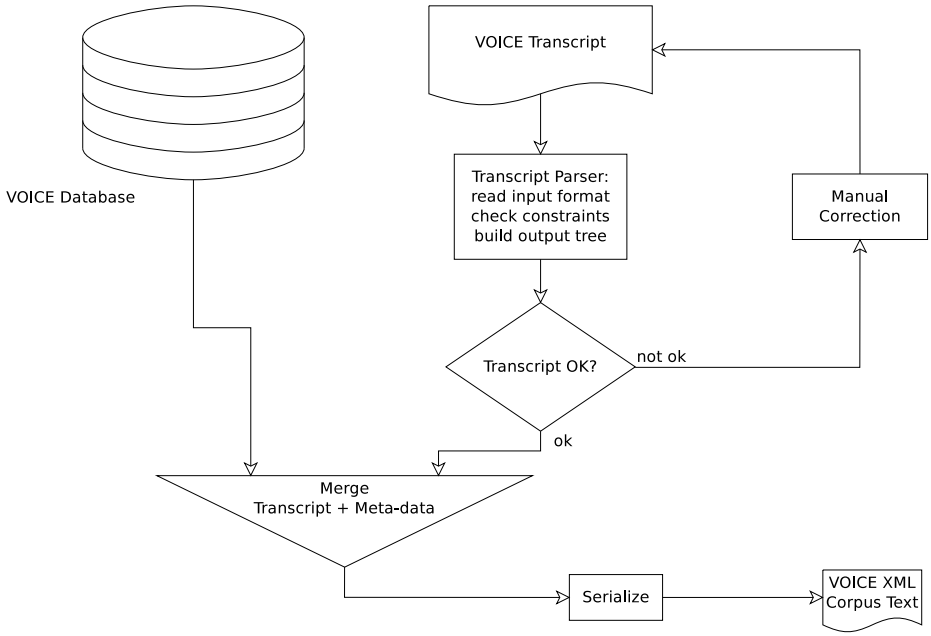



Figure 34: The full process from VOICE Transcript to the VOICE XML corpus text. The right-hand side of the flowchart shows the work-flow that is related to the transcripts. The data from the data-base on the left is merged after the parser has processed the input text.

to the speech-event of the transcript from the database and creates the suitable structures in the output format. These structures are then injected within the generated header (see table 8 Essential sections of the header and "`<teiHeader>`" (cf. Burnard & Bauman (eds.) 2008: section 2.1.1). Most notably, the information inserted at this stage are information on

- individual speakers
- language background of the speakers
- groups of speakers
- information on the publication
- stratification in 'domain' and 'speech-event type'
- power relations between speakers
- the speech event in prose
- the setting and location
- the used annotation in this particular file
- the responsibilities in the creation of the text
- the applicable license for the text

As a last step, the information is serialised into the resulting output tree. That means, the encoded data from the transcript and the data that has been retrieved from the meta-data database are written to a file. As the data is a tree-like structure this structure has to be serialised to a sequence of characters that can be stored as a file. The VOICE Parser

uses a standard-library implementing the XML document model. Therefore, the output-tree is in fact already an XML document. For this reason, the standard methods to the serialisation of XML can be used and it is therefore guaranteed that the result is a well-formed serialisation of XML. For this reason, it is possible to use VOICE XML with any available XML compliant processing-tools.

# 7 Conclusion and Outlook

The conclusion to this thesis is multifaceted, as this thesis includes discussions of several important and interlinked areas that are crucial for corpus-building. These range from attempting to approach the epistemic question of what annotation is – a question that will probably never be fully answered – to the discussion of more practical issues, such as how to provide a framework for the creation and maintenance of meta-data. Consequently, this conclusion aims to touch again on the distinct areas that are discussed in the individual chapters.

The declared goal of this thesis is to explain and document the design and implementation of a corpus resource, using VOICE as a real-life example. That means that it provides information on the main design decisions and explains some of the core aspects of corpus building from a technical perspective[42]. This case study is based on the work and research I have done for The Vienna-Oxford International Corpus of English. In this way, it explains and exemplifies how different scientific and technological decisions have to be matched. Parts of this thesis, therefore, might read like a reference manual. However, this is clearly intended, as this thesis describes the reasoning that is part of the creation history of a resource that is now publicly available (i.e. VOICE XML and VOICE Online).

The task for me, that is the goal of my research and the purpose of this thesis, was not only to find the right technologies supporting the needs of the researchers working in the field of English as a lingua franca. It was also a goal to establish a fruitful link to other disciplines, especially to computer sciences and digital humanities. From early on, this link proved to be a bi-directional connection that goes far beyond the mere provision of appropriate tools. On a practical level, it was a great benefit for the project to have custom-made IT tools that are tailored to the specific methodologies. On a conceptual level, the formalisms that were taken from other areas of research proved to be valuable[43] (e.g. the 'annotation graph', see *2.2. What is Annotation, Actually? p.7*).

The chapter on annotation (see *2. Annotation p.4*) introduces Bird & Liberman's (Bird & Liberman 2001) generic framework for linguistic annotations, which draws on abstract models from graph theory. An example that is illustrative of linguistic annotation, modeled

[42]The 'other' side, that is to say the design from a linguist point of view, including corpus balance and ethnographic dimensions, is more thoroughly described in Pitzl (2011)

[43]Some of these were drawn upon at conference talks and publications. See e.g. *Textbasierte Transkription in VOICE* (Majewski forthc.)

as an abstract annotation graph is discussed in detail to provide some understanding of the powerful nature of the concept. There, implications on its relation to transcription are discussed. Furthermore, the importance of annotation being theory neutral is introduced. That means that general rules are proposed to help to avoid a methodological mismatch between corpus builders and corpus users. The most important insights concerning theory neutral annotation are the preference of generally accepted annotations and the policy to be very explicit with the documentation of annotation that might be controversial or new. Furthermore, approaches to the temporal representation of spoken language are described. A system that distinguishes three temporal types of annotation (i.e eventive, modifying and temporally modifying) is proposed.

The chapters on transcription describe the path from pre-VOICE transcripts (see *3. The Legacy Format from the Pre-VOICE Era p.24*) to a transcription format that is formalised and can be structurally validated (*4. Redesigned Transcription Format p.29*). This part is mainly a documentation of decisions that were taken and an explanation on the reasoning behind these decisions. Topics that are specifically addressed are aspects that render VOICE transcripts readable and writable for human agents and formalisms that allow for unambiguous machine interpretation. This section highlights the benefits of text-based but still machine-readable transcription.

The chapter on the meta-data database (see *5. The VOICE Meta-data Database p.45*) highlights a particularly important area, that is the maintenance of meta-data. Therefore, it provides an explanation of the kind of information may be subsumed under the umbrella term 'meta-data'. Furthermore, this chapter highlights the areas of meta-data that are relevant for the annotation of VOICE and the kind of data that is crucial to administer the process of corpus creation.

The chapter on the corpus format (see *6. Corpus Publication Format p.55*) is most essential for a user of VOICE XML, but at the same time the most technical part of this thesis. This part can be read as complimentary documentation for the publicly available resource VOICE XML (VOICE Project 2011). Furthermore, it provides some historical background over the development of different corpus formats. With this, it is possible to see the position of the format of VOICE XML within the history of corpus building and its relation to other corpora. Furthermore, the selection of standards and co-standards is explained. The selection of standards described in this chapter mirror the decisions described in preceding parts of this thesis. The main focus of this chapter lies in a description of how the concepts encoded in VOICE transcripts are mapped to elements from the TEI eco-system.

The overall effort that went into VOICE can be regarded as a success, which is also echoed in the feedback of scientific community. What I learnt from the work on the project and within the scope of this thesis is that it can be very fruitful to bring together methodologies from different disciplines. Furthermore, we have seen that it can be a sound

decision to leave the mainstream paths of established methodology at some points and strive for innovation. The text-based transcription which is designed to be unambiguously computer-readable, human-readable and most importantly writable by human agents is one of these areas. This area is also illustrative of one of the fundamental qualities of the work for VOICE. That is to say that each member of the team had opportunity to include her/his share of expertise into the methodological framework. This thesis focuses on my area of expertise, but it is important to see and acknowledge that a project like VOICE would not have been possible without all other areas that are part of the methodology.

One of the most important conclusions is the following: When creating a digital resource for a humanities project, it is equally important to ask the question of *how* to do it technically, as it is to qualitatively define *what* to do. The two domains are seen in this thesis as facets of same intent. That is to say, they are serving a common goal and are mutually dependent. When *both* areas are converging, as it has been the case for VOICE, the result has the potential to be useful way beyond its original purpose. VOICE, in this case, might be useful for researchers outside of ELF. For example, the resource might be interesting for scientists working in the areas of natural language processing or researchers who are interested in spoken language from a more general perspective.

When taking the perspective of a user of VOICE, there would be much more that could be described in more detail. With this thesis, I tried to cover a great variety of interrelated but distinct areas. Of all parts that in total cover the topic *design and implementation*, only the most fundamental and most important concepts and decisions are documented. But certainly, as the range of different areas is wide and the confines of this thesis necessarily have to be limited, it is needless to say that for most of the covered areas, there would be more to elaborate on. That means that if a user of VOICE wants to use this text to learn on the specifics of the resource, there are certainly aspects that could have been explained more thoroughly. Nevertheless, the concepts as they are described here, should provide good guidance on how to approach VOICE from a technical perspective.

Many of the concepts and assets developed for VOICE on the basis of my work have a potential for further research. Especially the use of generative grammars for text-based transcripts appear to provide a sound basis for the application to other areas where data is encoded by transcribing experts. The use of a formally defined, text-based input format proved to be a very simple and flexible way to deal with complex types of data. The VoiceParser (see *VOICE Parser p.106*), which provides the transformation from transcript to XML, is a working proof-of-concept of computationally weaving together transcription and meta-data to the final XML corpus format. From the implementation side, the concept of transforming a text-based format could be refined to represent the syntax tree of the transcripts directly in the *XQuery 1.0 and XPath 2.0 Data Model (XDM)* (Berglund, Fernández, Malhotra, Marsh & Walsh 2010) to support more flexible tool chains. A great part of the knowledge that has been gathered within this thesis could be put to use to build

new tools for corpus builders as well as corpus users. This would certainly fill a gap. While various institutions put significant effort in the creation of open and interoperable standards for language data, the support of these standards by tools (i.e. computer programs) that are designed to create and use this kind of data could still be improved.

# Bibliography

AMI Project. 2009. *The AMI Meeting Corpus is a multi-modal data set consisting of 100 hours of meeting recordings*. http://corpus.amiproject.org/ (2010-09-13 06:59:10).

American National Corpus Project. 2009. *ANC Second Release*. http://www.americannationalcorpus.org/SecondRelease/ (2010-09-12 13:28:10).

Barras, Claude; Geoffrois, Edouard; Wu, Zhibiao; Liberman, Mark. 2001. "Transcriber: Development and use of a tool for assisting speech corpora production". *Speech Communication*. 33(1-2), 5-22.

Berglund, Anders; Fernández, Mary; Malhotra, Malhotra; Marsh, Jonathan; Walsh, Norman. 2010. *XQuery 1.0 and XPath 2.0 Data Model (XDM)*. http://www.w3.org/TR/xpath-datamodel/ (2011-07-24 20:48:03).

Bird, Steven; Liberman, Mark. 2001. "A formal framework for linguistic annotation". *Speech Communication*. 33(1-2), 23-60.

Boag, Scott; Chamberlin, Don; Fernández, Mary; Florescu, Daniela; Robie, Jonathan; Jérôme, Siméon. 2007. *XQuery 1.0: An XML Query Language*. http://www.w3.org/TR/xquery/ (2008-02-13 19:44:10).

Bray, Tim; Paoli, Jean; Maler, Eve; Yergeau, François. 2008. *Extensible Markup Language (XML) 1.0*. http://www.w3.org/TR/2008/REC-xml-20081126 (2009-09-11 14:35:43).

Breiteneder, Angelika. 2005. "Exploiting redundancy in English as a European lingua franca: the case of the 'third person -s"'. MA thesis, University of Vienna.

Breiteneder, Angelika; Pitzl, Marie-Luise; Majewski, Stefan; Klimpfinger, Theresa. 2006. "VOICE Recording-Methodological Challenges in the Compilation of a Corpus of Spoken ELF". *NJES: Nordic Journal of English Studies*. 5(2), 161-187.

Bronshtein, Ilja N.; Semendyayev, Konstantin A.; Musiol, Gerhard; Muehlig, Heiner. 2007. *Handbook of mathematics*. (fifth edition). Berlin: Springer.

Brown University. 2011. *TEI-L List at LISTSERV.BROWN.EDU*. http://listserv.brown.edu/archives/cgi-bin/wa?A0=TEI-L (2011-07-16 13:57:25).

Bruce, Gösta. 1992. "Design principles in the transcription of spoken discourse". In Svartvik, Jan (ed.). *Directions in corpus linguistics: proceedings of Nobel Symposium 82, Stockholm, 4-8 August 1991*.

Burnard, Lou. 1995. "What is SGML and how does it help?". *Computers & the Humanities*. 29(1), 41.

Burnard, Lou. 2002. "Where did we Go Wrong? A Retrospective Look at the British National Corpus". *Language and Computers*. 42, 51-70.

Burnard, Lou; Wynne, Martin (eds.). 2004. *BNC Baby*. (version 1.0). Oxford: Oxford University Computing Service.

Burnard, Lou. 2005. “Metadata for corpus work”. In Wynne, Martin (ed.). *Developing Linguistic Corpora: A Guide to Good Practice*. Oxford: Oxbow Books.

Burnard, Lou. 2007. *Reference Guide for the British National Corpus (XML Edition)*. http://www.natcorp.ox.ac.uk/docs/URG/ (2010-09-12 13:37:08).

Burnard, Lou; Bauman, Syd (eds.). 2008. *TEI P5: Guidelines for Electronic Text Encoding and Interchange*. http://www.tei-c.org/release/doc/tei-p5-doc/en/html/index.html (2008-02-14 13:38:21).

Carlisle, David; Ion, Patrick; Miner, Robert; Poppelier, Nico. 2003. *Mathematical Markup Language (MathML) Version 2.0*. http://www.w3.org/TR/MathML2/ (2010-08-29 13:23:30).

Clark, James. 1997. *Comparison of SGML and XML*. http://www.w3.org/TR/NOTE-sgml-xml-971215 (2010-08-29 10:59:48).

Clark, James; Lipkin, Daniel; Marsh, Jonathan; Thompson, Henry; Walsh, Norman; Zilles, Steve. 1999. *XSL Transformations (XSLT)*. http://www.w3.org/TR/xslt (2010-08-28 15:07:45).

Collins, Peter; Hollo, Carmella. 2000. *English grammar: an introduction*. (edition 2). London: Macmillan.

Dahlström, Erik; Ferraiolo, Jon; Fujisawa, Jun; Jackson, Dean; Lilley, Chris; McCormack, Cameron; Schepers, Doug; Watt, Jonathan; Dengler, Patrick. 2010. *Scalable Vector Graphics (SVG) 1.1*. http://www.w3.org/TR/SVG11/ (2010-08-29 13:29:24).

Digital Equipment Corporation. 1992. *Database Language SQL*. http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt (2011-03-21 15:53:04).

Dipper, Stefanie. 2005a. “XML-based stand-off representation and exploitation of multi-level linguistic annotation”. *Proceedings of Berliner XML Tage*. Berlin.

Dipper, Stefanie; Götze, Michael; Stede, Manfred (eds.). 2005b. *Heterogeneity in Focus: Creating and Using Linguistic Databases*. Universitätsverlag Potsdam.

Dipper, Stefanie; Hinrichs, Erhard; Schmidt, Thomas; Wagner, Andreas; Witt, Andreas. 2006. “Sustainability of Linguistic Resources”. *Proceedings of the LREC Workshop on merging and layering linguistic information*. Genoa.

Dodd, Tony. 2010. *XAIRA (XML aware indexing and retrieval architecture)*. Version 1.26.

Du Bois, John W.; Schuetze-Coburn, Stephan; Cumming, Susanna; Paolino, Danae. 1993. “Outline of discourse transcription”. In Edwards, Jane Anne; Lampert, Martin D. (eds.). *Talking data: Transcription and coding in discourse research*. Hillsdale NJ: Lawrence Erlbaum Associates.

Dunlop, Dominic. 1995. “Practical considerations in the use of TEI headers in a large corpus.”. *Computers & the Humanities*. 29(1), 85.

Edwards, Jane Anne. 1992. “Design principles in the transcription of spoken discourse”. In Svartvik, Jan (ed.). *Directions in corpus linguistics: proceedings of Nobel Symposium 82, Stockholm, 4-8 August 1991*.

Edwards, Jane Anne; Lampert, Martin D. (eds.). 1993. *Talking data: transcription and coding in discourse research*. Routledge.

Fallside, David; Walmsley, Priscilla. 2004. *XML Schema Part 0: Primer*. http://www.w3.org/TR/xmlschema-0/ (2010-08-29 16:16:23).

Farrar, Scott; Lewis, William D. 2007. "The GOLD Community of Practice: an infrastructure for linguistic data on the Web". *Language Resources and Evaluation*. 41(1), 45-60.

Francis, Nelson W.; Kucera, Henry. 1979. *Brown Corpus Manual*. (edition 3). Providence, Rhode Island: Brown University.

Garside, Roger; Leech, Geoffrey N.; McEnery, Tim. 1997. *Corpus annotation: linguistic information from computer text corpora*. London: Longman.

Hornby, Albert; Wehmeier, Sally; McIntosh, Colin; Turnbull, Joanna; Ashby, Michael. 2005. *Oxford Advanced Learner's Dictionary*. (edition 7). Oxford: Oxford University Press.

International Organisation for Standardisation (ISO). 1986. "ISO 8879-1986, Information processing - Text and Office Systems - Standard Generalized Markup Language (SGML)". *ISO Standards*.

International Organisation for Standardisation (ISO). 2004. "ISO 8601:2004, Data elements and interchange formats – Information interchange – Representation of dates and times". *ISO Standards*.

Kilgour, Jonathan. 2008. *NITE XML Toolkit - Documentation*. http://groups.inf.ed.ac.uk/nxt/documentation.shtml (2008-02-18 08:46:33).

Klimpfinger, Theresa. 2005. "The role of speakers' first and other languages in English as a lingua franca talk". MA thesis, University of Vienna.

Knuth, Donald E. 1964. "Backus normal form vs. backus naur form". *Communications of the ACM*. 7(12), 735–736.

Kordon, Kathrin. 2003. "Phatic communion in English as a lingua franca". MA thesis, University of Vienna.

Lee, James; Ware, Brent. 2003. *Open source Web development with LAMP: using Linux, Apache, MySQL, Perl, and PHP*. Boston: Addison-Wesley.

Majewski, Stefan. 2010. *VoiceScribe*. Version 1.0.2. Vienna.

Majewski, Stefan. (forthc.). "Textbasierte Transkription in VOICE". In Wandl-Vogt, Eveline; Korecky-Kröll, Katharina (eds.). *Tagungsband des Zentrums für Sprachwissenschaft, Bild- und Tondokumentation der österreichischen Akademie der Wissenschaften zum Thema "Transkriptionssysteme: Sprache - Bild - Ton. Codierung gesprochener Sprache"*. Vienna: Praesens.

Marsh, Jonathan; Orchard, David; Veillard, Daniel. 2006. *XML Inclusions (XInclude) Version 1.0*. http://www.w3.org/TR/2006/REC-xinclude-20061115/ (2010-08-29 16:38:56).

Mauranen, Anna. 2006. "A Rich Domain of ELF-the ELFA Corpus of Academic Discourse". *NJES: Nordic Journal of English Studies*. 5(2), 145-159.

Mazzoni, Dominic. 2006. *Audacity*. Version 1.2.x.

Michigan Corpus Linguistics. 2010. *Michigan Corpus of Academic Spoken English » ELI Corpora & UM ACL*. http://micase.elicorpora.info/ (2010-09-12 13:42:58).

Mitton, Roger; Hardcastle, David; Pedler, Jenny. 2007. *BNC! Handle with care! Spelling and tagging errors in the BNC*. London: Birkbeck ePrints.

Newman, Simon M.; Swanson, Rowena W.; Knowlton, Kenneth C. 1959. "A Notation System for Transliterating Technical and Scientific Text for Use in Data Processing Systems". *Advances in Documentation and Library Science*. 3(1), 345–376.

O'Neil, Patrick; O'Neil, Elizabeth. 2001. *Database–principles, programming, and performance*. San Francisco: Morgan Kaufmann.

Oracle. 2010. *MySQL: The world's most popular open source database*. http://mysql.com/ (2011-03-22 11:20:05).

Oracle. 2011. *About OpenOffice.org*. http://about.openoffice.org/ (2011-02-25 15:17:46).

Osimk, Ruth. (forthc.). "Evaluating the applicability of existing POS practices for a corpus of English as a lingua franca (VOICE)". In Hoffmann, Sebastian; Rayson, Paul; Leech, Geoffrey (eds.). *Corpus linguistics and variation in English: Focus on Non-native Englishes (Proceedings of ICAME 30)*. Helsinki: Research Unit for Variation, Contacts and Change in English (VARIENG), University of Helsinki.

Oxford University Computing Services. 2009. *British National Corpus*. http://www.natcorp.ox.ac.uk/ (2010-09-12 13:33:58).

Phillips, Addison; Davis, Mark. 2009. *BCP47: Tags for Identifying Languages*. http://www.rfc-editor.org/rfc/bcp/bcp47.txt (2011-03-30 09:45:24).

Pitzl, Marie-Luise. 2004. "'I know what you mean' – 'miscommunication' in English as a lingua franca: the case of business meetings". MA thesis, University of Vienna.

Pitzl, Marie-Luise; Breiteneder, Angelika; Klimpfinger, Theresa. 2008. "A world of words: processes of lexical innovation in VOICE". *Vienna English Working PaperS*.

Pitzl, Marie-Luise. 2011. "Creativity in English as a lingua franca: Idiom and metaphor". PhD thesis, University of Vienna.

Quin, Liam. 2009. *The Extensible Stylesheet Language Family (XSL)*. http://www.w3.org/Style/XSL/ (2010-08-28 15:12:34).

Rahtz, Sebastian. 2007. *P4 to P5 converter*. Oxford: TEIConsortium.

Sampson, Geoffrey; McCarthy, Diana. 2005. *Corpus Linguistics: Readings in a Widening Discipline*. Continuum International Publishing Group.

Schmidt, T. 2001. “The transcription system EXMARaLDA: An application of the annotation graph formalism as the Basis of a Database of Multilingual Spoken Discourse”. *Proceedings of the IRCS Workshop on Linguistic Databases*.

Scott, Mike. 2010. *WordSmith Tools*. Version 5.0. Oxford University Press.

Seidlhofer, Barbara. 2001. “Closing a Conceptual Gap: The Case for a Description of English as a Lingua Franca”. *International Journal of Applied Linguistics*. 11(2), 133-158.

Seidlhofer, Barbara. 2005. “English as Lingua Franca”. *ELT Journal*. 59(4), 339-341.

Seidlhofer, Barbara; Breiteneder, Angelika; Pitzl, Marie-Luise. 2006. “English as a lingua franca in Europe: challenges for applied linguistics”. *Annual Review of Applied Linguistics*. 26, 3–34.

Simpson, Rita C.; Lee, David Y. W.; Leicher, Sheryl; Ädel, Annelie (eds.). 2007. *The MICASE handbook: a resource for users of the Michigan corpus of academic spoken English*. (edition 3 work in progress). Ann Arbor: University of Michigan.

Sinclair, John McHardy; Carter, Ronald. 2004. *Trust the Text: Language, Corpus and Discourse*. Routledge.

Sperberg-McQueen, C. M.; Huitfeldt, C.; Renear, A. 2000. “Meaning and interpretation of markup”. *Markup Languages Theory and Practice*. 2(3), 215–234.

Strasser, Thomas. 2004. “The use of English as a Lingua Franca in a large Austrian company”. MA thesis, University of Vienna.

Svartvik, Jan (ed.). 1992. *Directions in Corpus Linguistics: Proceedings of Nobel Symposium 82, Stockholm, 4-8 August 1991*. Mouton De Gruyter.

TEI-Consortium. 2007. *Guidelines for Electronic Text Encoding and Interchange*. http://www.tei-c.org/Guidelines/P5 (2009-11-19 23:00:00).

Unicode Consortium. 2010. *Unicode 5.2.0*. http://www.unicode.org/versions/Unicode5.2.0/ (2010-09-13 10:25:57).

VOICE. 2009. *The Vienna-Oxford International Corpus of English*. http://voice.univie.ac.at (2009-11-26 14:24:54).

VOICE Project. 2007a. *Mark-up Conventions*. http://www.univie.ac.at/voice/documents/VOICE_mark-up_conventions_v2-1.pdf (2010-07-13 11:30:59).

VOICE Project. 2007b. *Spelling Conventions*. http://www.univie.ac.at/voice/documents/VOICE_spelling_conventions_v2-1.pdf (2010-07-13 11:30:59).

VOICE Project. 2008. *VOICE website*. http://www.univie.ac.at/voice/ (2008-02-16 07:51:44).

VOICE Project. 2009a. *VOICE - Availability*. http://www.univie.ac.at/voice/page/corpus_availability (2009-11-26 14:27:03).

VOICE Project. 2009b. *VOICE - VOICE Online Statistics*. http://www.univie.ac.at/voice/stats (2011-03-01 11:09:06).

VOICE Project. 2009c. *The Vienna-Oxford International Corpus of English*. (version 1.0 online).

VOICE Project. 2009d. *VOICE - Terms of Use*. http://www.univie.ac.at/voice/page/terms_of_use (2011-07-01 17:35:54).

VOICE Project. 2011. *The Vienna-Oxford International Corpus of English*. (version 1.1 XML).

Widdowson, Henry G. 2006. "Applied linguistics and interdisciplinarity". *International Journal of Applied Linguistics*. 16(1), 93-96.

Xiao, Richard. 2006. "REVIEW: Xaira – an XML aware indexing and retrieval architecture.". *Corpora*. 1(1), 99-104.

de Castro Lopo, Erik. 2009. *libsndfile*. Version 1.0.18.

# Appendix

## A Abbreviations

**ANC** American National Corpus

**BNC** British National Corpus

**DTD** Document Type Definition

**IPA** International Phonetic Alphabet

**L1** The first language of a speaker

**PVC** Pronunciation, variation and coinage.
The threshold from which an expression is regarded a PVC is established by referring to a reference dictionary, the 7th edition of the *Oxford Advanced Learner's Dictionary* (OALD) (Hornby, Wehmeier, McIntosh, Turnbull & Ashby 2005) and by rules based on the count of syllables in a word deviating from this reference (see Pitzl, Breiteneder & Klimpfinger 2008).

**RAM** Random Access Memory

**SGML** Standard Generalized Markup Language.

**TEI** The Text Encoding Initiative is an initiative that seeks to establish common guidelines for the encoding of text. These guidelines, today, propose an XML based data format. The abbreviation 'TEI' is commonly used to refer both to the initiative, that is the TEI Consortium, and the proposed guidelines. The guidelines are published in proposals. The current major edition is 'P5'.

**VPN** Virtual Private Network. A network that establishes a safe private connection between computers over a non-safe public network. The computers can, independent from their location, exchange data with each other as if they were directly connected with network cables.

**white-space** Any non-printing spacing characters. Examples are, among others, single spaces, non-breaking space, tabulator characters or newline characters.

**XML** Extensible Markup Language

# B Software developed and used for VOICE

The following parts of the appendix are intended to give an overview of the software I developed within the VOICE project. This is included in this thesis as these are implementations that are based on the concepts described in the main part of the text. This appendix is followed by another appendix with a selection of program code (see *Code Developed for VOICE p.108*).

## B.1 VOICE Parser

The VOICE Parser is the reference implementation of the annotation and mark-up scheme devised for VOICE. It is used to read and validate transcripts based on the VOICE *Mark-up Conventions* (see *VOICE Transcription Conventions p.219*) and transform them to the TEI-based XML corpus format.

The parser performs three different steps. First of all, it provides parser rules to detect and read VOICE mark-up. Secondly, it builds a representation of the encoded information with the corresponding structures in the target corpus format. Finally, it merges data from a database with the annotated texts.

## B.2 VoiceScribe

*VoiceScribe* (Majewski 2010) is a simple and easy to use transcript editor. The idea to create a custom editor evolved while the team was busy designing the *VOICE Transcription Conventions*. It features an in-program audio player, a pattern based annotation highlighting editor with basic text search functionality. VoiceScribe saves its transcripts in plain text format.

### B.2.1 Text Highlighting

The most important aspect of VoiceScribe is the highlighting of the mark-up. While a transcriber works on a transcript, immediate visual feedback on the correctly recognised mark-up features is provided. VoiceScribe does not completely model the transcription format and is therefore not in the strict sense a validator of the mark-up. Nevertheless, it helps to eliminate the common formal inconsistencies and reduces the time a transcriber has to spend on correcting and proofing the transcripts. Despite this, it is important to note that while formal consistency is important for the further processing, the manual effort to check the transcripts for the qualitative consistency of the annotation requires the greatest part of the transcriber's attention. For this reason VoiceScribe assists with the formal validity of the mark-up, so that the transcriber can work more focused on the qualitative part of the work that requires his/her expert knowledge.

### B.2.2 Audio Component

The audio player component of VoiceScribe is based on the audio playback engine the open-source audio editor *Audacity* (Mazzoni 2006). The current version of VoiceScribe supports the WAV and FLAC lossless audio formats. Lossy formats like MP3 (MPEGII layer 3) are currently not supported. As the used playback component reads files with libsndfile (de Castro Lopo 2009), all file formats this library supports can be enabled for playback in the program code.

The main operations supported by the audio component are:

- linear playback
- repetitive playback of a portion of audio
- continuation to the next portion after a configurable number of repetitions
- manual repetition
- manual jump to the next portion
- configurable pause between repetitions
- pause/start playback

The commands of the audio components are all accessible via keyboard short-cuts. Thus, it is not necessary for a transcriber to move the hands off the keyboard to control the audio component. When a transcriber is fully accustomed to the keyboard-based control, this reduces the strain the operation of the software imposes on the transcriber.

# C Code Developed for VOICE

Many of the conceptual decisions for the encoding of VOICE have a representation in different kinds of computer code. The following sections within this appendix provide the most essential code fragments for reference.

## C.1 VOICE Parser

The following code is the program that transforms VOICE transcripts to the XML corpus format (see *6. Corpus Publication Format p.55*). Furthermore it merges the content of the meta-data database (see *5. The VOICE Meta-data Database p.45*). It is written in the *Perl* programming language, which is an interpreted scripting language.

```
1  #!/usr/bin/perl
2  #
3  # author: Stefan Majewski
4  #
5  # project: VoiceParser
6  #
7  # description: A Parser for the transformation of VOICE
8  #   transcripts to TEI XML. Some post-processing of the generated
9  #   files is required to match the corpus format of VOICE 1.1 XML
10 #
11 # date completed: 2009
12
13 use strict;
14 use warnings;
15 use XML::Xerces;
16 use Getopt::Long;
17 use utf8;
18 use encoding 'utf8';
19 use DBD::mysql;
20 use Date::ISO;
21
22 our $it = '';
23
24
25 #
26 # Initialisation and cleanup of package
27 #
28 BEGIN{
29     XML::Xerces::XMLPlatformUtils::Initialize();
30       $SIG{__WARN__} = sub {
31           $it->status($_[0]) if defined $^S;
32       }
33 }
34 END{
35     XML::Xerces::XMLPlatformUtils::Terminate();
36 }
37
38 ######################################################################
39 # <declarations>
40 # actually not needed in package, but provided as an index
41 ######################################################################
42
43 package VoiceParser;
44
45 sub parse_transcript(\$$);
46 sub parse_header(\$$);
```

```
47  sub parse_body(\$$);
48  sub parse_transcriber_notes(\$$);
49  sub parse_tn_hash(\$$);
50  sub parse_tn_excl(\$$);
51  sub parse_tn_text(\$$);
52  sub parse_u (\$$);
53  sub parse_sid (\$$);
54  sub parse_space (\$$);
55  sub parse_wpu (\$$);
56  sub parse_wpl (\$$);
57  sub parse_wph (\$$);
58  sub parse_wplaugh (\$$);
59  sub parse_ipa_wp(\$$);
60  sub parse_pause (\$$);
61  sub parse_length (\$$);
62  sub parse_rise (\$$);
63  sub parse_fall (\$$);
64  sub parse_uncertain (\$$);
65  sub parse_xl(\$$);
66  sub parse_xu(\$$);
67  sub parse_unintell (\$$);
68  sub parse_contextual (\$$);
69  sub parse_punct(\$$);
70  sub parse_you_dis_cls(\$$);
71  sub parse_lx(\$$);
72  sub parse_sm_laugh(\$$);
73  sub parse_anon(\$$);
74  sub parse_hash(\$$);
75  sub parse_excl(\$$);
76  sub parse_track(\$$);
77  sub parse_spel(\$$);
78  sub parse_sn(\$$);
79  sub parse_sm(\$$);
80  sub parse_ol(\$$);
81  sub parse_pvc(\$$);
82  sub parse_wpyear(\$$);
83  sub parse_othercont(\$$);
84  sub parse_directed(\$$);
85  sub parse_ono(\$$);
86
87  ### logic groupings
88  sub parse_wp(\$$);
89  sub parse_wp_nw(\$$);
90  sub parse_wp_stretch(\$$);
91
92  ### helper functions
93  sub eat_nl(\$);
94
95  ######################################################################
96  # </declarations>
97  ######################################################################
98
99
100
101
102 ######################################################################
103 # <methods/routines>
104 ######################################################################
105
106 #
107 # creation of new instance, initialisation of all required vars
108 #
109 sub new(){
```

```
110     my ($this) = @_;
111     my $class = ref($this) || 'VoiceParser';
112     my $self = {DEBUG => 0,
113                 NO_WORD => 1,
114                 UTTERANCE_CNT => 0,
115                 SHIFT_CNT => 0,
116                 HAVE_SID_IN_PARTICDESC => {},
117                 OVERLAP_CNT => 0,
118                 OVERLAP_IN_UTTERANCE => 0,
119                 OVERLAP_MAP => {},
120                 OVERLAP_REFNUM => {},
121                 REVISION => {},
122                 REVDATE => {},
123                 REVNUM => {},
124                 REVCNT => 0,
125                 DISAMBIGUATE_OVERLAPS => 1,
126                 DISAMBIGUATE_OVERLAPS_DEPTH => 6,
127                 FRESH_UTTERANCE => 0,
128                 FINISHED_UTTERANCE => 0,
129                 OTHER_CONT_CNT => 0,
130                 OTHER_CONT_B_CNT => 0,
131                 OTHER_CONT_C_CNT => 0,
132                 XMLID_PREFIX => 'vtr',
133                 LINE_CNT => 1,
134                 DOC_TEXT => '',
135                 DOC_HEADER => '',
136                 DOC_ROOT => '',
137                 DOC_NOTES => '',
138                 DOC => '',
139                 DB_H => '',
140                 DB_HOST => 'localhost',
141                 DB_NAME => '',
142                 DB_USER => '',
143                 DB_PASS => '',
144                 USE_DB => '',
145                 IMPL => '',
146                 H_SHORT_TITLE => ''
147                 };
148     bless $self, $class;
149     $it = $self;
150     $self->initialize();
151     return $self;
152 }
153
154 #
155 # The XML related initialisation
156 #
157 sub initialize(){
158     my ($self) = @_;
159     $self->{IMPL} = XML::Xerces::DOMImplementationRegistry::...
        ...getDOMImplementation('LS');
160     my $dt = $self->{IMPL}->createDocumentType('VOICE', '', '');
161     $self->{DOC} = $self->{IMPL}->createDocument('http://www.tei-c.org/ns/1.0'...
        ..., 'TEI', $dt);
162     my $d = $self->{DOC};
163     $self->{DOC_ROOT} = $d->getDocumentElement();
164     $self->{DOC_ROOT}->setAttribute('version', '5');
165 #     $self->{DOC_ROOT}->setAttribute('xmlns', 'http://www.tei-c.org/ns/1.0');
166     $self->{DOC_ROOT}->setAttribute('xmlns:voice', 'http://www.univie.ac.at/...
        ...voice/ns/1.0');
167     $self->{DOC_ROOT}->setAttribute('xml:lang', 'eng');
168     $self->{DOC_ROOT}->setAttribute('xmlns:xi', 'http://www.w3.org/2001/...
        ...XInclude');
```

```
169       $self->{DOC_TEXT} = $d->createElement('text');
170       $self->{DOC_BODY} = $d->createElement('body');
171       $self->{DOC_FILEDESC} =  $d->createElement('fileDesc');
172       $self->{DOC_ENCODINGDESC} = $d->createElement('encodingDesc');
173       $self->{DOC_PROFILEDESC} = $d->createElement('profileDesc');
174       $self->{DOC_REVISIONDESC} = $d->createElement('revisionDesc');
175       $self->{DOC_HEADER} = $d->createElement('teiHeader');
176       $self->{DOC_HEADER}->appendChild($self->{DOC_FILEDESC});
177       $self->{DOC_HEADER}->appendChild($self->{DOC_ENCODINGDESC});
178       $self->{DOC_HEADER}->appendChild($self->{DOC_PROFILEDESC});
179       $self->{DOC_HEADER}->appendChild($self->{DOC_REVISIONDESC});
180       $self->{DOC_ROOT}->appendChild($self->{DOC_HEADER});
181       $self->{DOC_ROOT}->appendChild($self->{DOC_TEXT});
182       $self->{DOC_TEXT}->appendChild($self->{DOC_BODY});
183       $self->{DOC_TRANSNOTES} = $d->createDocumentFragment();
184       $self->{EDITION} = "VOICE␣1.0␣Online";
185       $self->{PUBLISHER} = "Vienna-Oxford␣International␣Corpus␣of␣English,␣...
              ...University␣of␣Vienna";
186       $self->{DISTRIBUTOR} = "Vienna-Oxford␣International␣Corpus␣of␣English,␣...
              ...University␣of␣Vienna";
187       $self->{DOMAIN} = '';
188       $self->{SPET} = '';
189   #     $self->{ADDRESS} = 'Spitalgasse 2-4, 1090 Vienna, Austria, +43 1 ...
          ...427747447, voice@univie.ac.at, http://www.univie.ac.at/voice';
190   }
191
192   #
193   # helper method: print debug messages
194   #
195   sub debug($){
196       my ($self, $debug) = @_;
197       $self->{DEBUG} = $debug;
198       return $debug;
199   }
200
201   #
202   # helper method: normalisation of subsequent spaces.
203   #
204   sub normalize_space($){
205       my ($self, $str) = @_;
206       $str =~ s/ /_/g;
207       return $str;
208   }
209
210   #
211   # helper method: set the distance ,in subsequent overlaps, from
212   # which two subsequent overlaps are regarded to be unrelated
213   #
214   sub set_disambiguation_depth($){
215       my ($self, $depth) = @_;
216       $self->{DISAMBIGUATE_OVERLAPS_DEPTH} = $depth;
217       warn "set␣disambiguationdepth␣to␣$depth";
218       return $depth;
219   }
220
221   #
222   # helper method: sets the text id for a transcript
223   #
224   sub set_id_prefix($){
225       my ($self, $prefix) = @_;
226       $self->{XMLID_PREFIX} = $prefix;
227       $self->{DOC_ROOT}->setAttribute('xml:id', $prefix);
228   }
```

```
229
230 #
231 # helper method: set the id from the database (preferred over
232 # set_id_prefix).
233 #
234 sub set_id_prefix_db($){
235     my ($self) = @_;
236     if($self->{USE_DB}){
237         unless($self->{DB_H}){
238             $self->connect_db();
239         }
240     }
241 #     my $sh = $self->{DB_H}->prepare("SELECT xml_prefix, short_title FROM ...
    ...events WHERE short_title = '" . $self->{H_SHORT_TITLE} . "'");
242     my $sh = $self->{DB_H}->prepare("SELECT␣corpus_spets.xml_id_comp␣AS␣spet,␣...
        ...corpus_domains.xml_id_comp␣AS␣domain,␣CONCAT(corpus_domains....
        ...xml_id_comp,␣corpus_spets.xml_id_comp,␣ID)␣AS␣xml_prefix␣from␣events...
        ...,␣corpus_domains,␣corpus_spets␣WHERE␣events.corp_domain␣=␣...
        ...corpus_domains.domain_id␣AND␣events.corp_spet␣=␣corpus_spets.spet_id...
        ...␣AND␣short_title␣=␣'" . $self->{H_SHORT_TITLE} . "'");
243     $sh->execute();
244     my $r = $sh->fetchrow_hashref();
245
246     if($r){
247
248         if($r->{'spet'}){
249             $self->{SPET} = $r->{'spet'};
250         }
251         if($r->{'domain'}){
252             $self->{DOMAIN} = $r->{'domain'};
253         }
254
255         if($r->{'xml_prefix'}){
256             $self->set_id_prefix($r->{'xml_prefix'});
257         }
258         else{
259             $self->set_id_prefix($self->{H_SHORT_TITLE});
260             warn "No␣xml_prefix␣in␣database:␣using␣short_title␣instead";
261         }
262     }
263     else{
264             $self->set_id_prefix($self->{H_SHORT_TITLE});
265             warn "Could␣not␣set␣xml_prefix␣from␣database:␣using␣short_title␣...
                ...instead";
266     }
267 }
268
269 #
270 # helper method: prints a doctype (obsolete)
271 #
272 sub print_doctype(){
273     my ($self) = @_;
274     print <<EOD;
275 <!DOCTYPE voice [
276 <!ENTITY len ":">
277 <!ENTITY rise "?">
278 <!ENTITY  fall ".">
279 ]>
280 EOD
281 #STDERR->autoflush(1);
282 $|=1;
283 }
284
```

```
285 #
286 # helper method: serialize the Document Object Model (DOM) --> XML
287 #
288 sub write_xml($$){
289     my ($self, $path, $pretty) = @_;
290 #     $self->print_doctype();
291     my $dw = $self->{IMPL}->createDOMWriter();
292     if($pretty){
293         if($dw->canSetFeature('format-pretty-print',1)){
294             $dw->setFeature('format-pretty-print', 1);
295         }
296     }
297     warn("Saving as " . $self->{XMLID_PREFIX} . '.xml');
298     my $out = XML::Xerces::LocalFileFormatTarget->new(($path?$path:'.') . '/' ...
        .... $self->{XMLID_PREFIX} . '.xml');
299     $dw->writeNode($out, $self->{DOC_ROOT});
300 }
301
302 #
303 # helper method: output XML
304 #
305 sub print_xml($){
306     my ($self, $pretty) = @_;
307     $self->print_doctype();
308     my $dw = $self->{IMPL}->createDOMWriter();
309     if($pretty){
310         if($dw->canSetFeature('format-pretty-print',1)){
311             $dw->setFeature('format-pretty-print', 1);
312         }
313     }
314     my $out = XML::Xerces::StdOutFormatTarget->new();
315     $dw->writeNode($out, $self->{DOC_ROOT});
316 }
317
318 #
319 # helper: print a status message with reference to the input line
320 #
321 sub status($){
322     my ($self, $msg) = @_;
323     print STDERR $self->{LINE_CNT} . ": " . $msg . "\n";
324 }
325
326 ######################################################################
327 # database related methods
328 ######################################################################
329
330 #
331 # connects to the database with meta-data
332 #
333 sub connect_db (){
334     my ($self, $d, $h, $u, $p) = @_;
335     $d = $self->{DB_NAME} unless $d;
336     $h = $self->{DB_HOST} unless $h;
337     $u = $self->{DB_USER} unless $u;
338     $p = $self->{DB_PASS} unless $p;
339     $self->{DB_H} = DBI->connect("DBI:mysql:database=${d};host=${h}",
340                                  $u, $p,
341                                  {RaiseError => 1});
342
343 }
344
345 #
346 # sets the connection information for the database-connection
```

```
347 #
348 sub set_db($$$$){
349     my ($self, $d, $h, $u, $p) = @_;
350     $self->{DB_NAME} = $d;
351     $self->{DB_HOST} = $h;
352     $self->{DB_USER} = $u;
353     $self->{DB_PASS} = $p;
354     $self->{USE_DB} = 1;
355 }
356
357 #
358 # generates the header information. Pulls data from the meta-data
359 # database
360 #
361 sub generate_header (){
362     my ($self) = @_;
363     if($self->{USE_DB}){
364         unless($self->{DB_H}){
365             $self->connect_db();
366         }
367     }
368     my $d = $self->{DOC};
369
370
371     $self->generate_encoding_desc();
372
373     # fileDesc
374     # some further initialisation
375     $self->{DOC_PARTICDESC} = $d->createElement('particDesc');
376     # $self->{DOC_TITLE_STMT} = $d->titleStmt('titleStmt');
377     $self->{DOC_NOTES} = $d->createElement('notesStmt');
378     $self->{DOC_SOURCEDESC} = $d->createElement('sourceDesc');
379
380
381     if($self->{USE_DB} && $self->{DB_H}){
382     # title
383         my $sh = $self->{DB_H}->prepare("SELECT * FROM events WHERE ...
            ... short_title = '" . $self->{H_SHORT_TITLE} . "'");
384         $sh->execute();
385         if(my $r = $sh->fetchrow_hashref()){
386             my $n1 = '';
387             my $n2 = '';
388             $n1 = $d->createElement('titleStmt');
389             $n2 = $d->createElement('title');
390             $n2->appendChild($d->createTextNode($r->{'corp_title'}));
391             $n1->appendChild($n2);
392             $n2 = $d->createElement('principal');
393             $n2->appendChild($d->createTextNode('Prof. Barbara Seidlhofer'));
394             $n1->appendChild($n2);
395             $n2 = $d->createElement('editor');
396             $n1->appendChild($n2);
397             $n2 = $d->createElement('funder');
398             $n2->appendChild($d->createTextNode('Austrian Science Foundation (...
                ...FWF)'));
399             $n1->appendChild($n2);
400             $n2 = $d->createElement('sponsor');
401             $n1->appendChild($n2);
402             $n2 = $d->createElement('respStmt');
403             my $n3 = $d->createElement('resp');
404             $n3->appendChild($d->createTextNode('Corpus Compilation'));
405             $n2->appendChild($n3);
406             $n3 = $d->createElement('orgName');
407             $n3->appendChild($d->createTextNode('VOICE Project'));
```

```
408             $n2->appendChild($n3);
409             $n3 = $d->createElement('persName');
410             $n3->appendChild($d->createTextNode('Barbara Seidlhofer'));
411             $n2->appendChild($n3);
412             $n3 = $d->createElement('persName');
413             $n3->appendChild($d->createTextNode('Angelika Breiteneder'));
414             $n2->appendChild($n3);
415             $n3 = $d->createElement('persName');
416             $n3->appendChild($d->createTextNode('Marie-Luise Pitzl'));
417             $n2->appendChild($n3);
418             $n3 = $d->createElement('persName');
419             $n3->appendChild($d->createTextNode('Stefan Majewski'));
420             $n2->appendChild($n3);
421             $n3 = $d->createElement('persName');
422             $n3->appendChild($d->createTextNode('Theresa Klimpfinger'));
423             $n2->appendChild($n3);
424             $n1->appendChild($n2);
425             $self->{DOC_FILEDESC}->appendChild($n1);
426
427             $n1 = $d->createElement('editionStmt');
428             $n2 = $d->createElement('edition');
429             $n2->appendChild($d->createTextNode($self->{EDITION}));
430             $n1->appendChild($n2);
431             $self->{DOC_FILEDESC}->appendChild($n1);
432
433             $n1 = $d->createElement('extent');
434             $self->{DOC_EXTENT} = $n1;
435             $n1->appendChild($d->createTextNode('                    '));
436             $self->{DOC_FILEDESC}->appendChild($n1);
437
438             $n1 = $d->createElement('publicationStmt');
439             $n2 = $d->createElement('publisher');
440             $n2->appendChild($d->createTextNode($self->{PUBLISHER}));
441             $n1->appendChild($n2);
442             $n2 = $d->createElement('distributor');
443             $n2->appendChild($d->createTextNode($self->{DISTRIBUTOR}));
444             $n1->appendChild($n2);
445             $n2 = $d->createElement('address');
446             $n3 = $d->createElement('addrLine');
447             $n3->appendChild($d->createTextNode('Spitalgasse 2-4'));
448             $n2->appendChild($n3);
449             $n3 = $d->createElement('addrLine');
450             $n3->appendChild($d->createTextNode('1090 Vienna'));
451             $n2->appendChild($n3);
452             $n3 = $d->createElement('addrLine');
453             $n3->appendChild($d->createTextNode('Austria'));
454             $n2->appendChild($n3);
455             $n3 = $d->createElement('addrLine');
456             $n3->appendChild($d->createTextNode('+43 1 427742446'));
457             $n2->appendChild($n3);
458             $n3 = $d->createElement('addrLine');
459             $n3->appendChild($d->createTextNode('voice@univie.ac.at'));
460             $n2->appendChild($n3);
461             $n1->appendChild($n2);
462             $n3 = $d->createElement('addrLine');
463             $n3->appendChild($d->createTextNode('http://www.univie.ac.at/voice...
                    ...'));
464             $n2->appendChild($n3);
465             $n2 = $d->createElement('xi:include');
466             $n2->setAttribute('href', '../include/availability.xml');
467             $n2->setAttribute('parse', 'xml');
468             $n1->appendChild($n2);
469             $n2 = $d->createElement('idno');
```

```
470             $n2->appendChild($d->createTextNode($self->{XMLID_PREFIX}));
471             $n1->appendChild($n2);
472             $self->{DOC_FILEDESC}->appendChild($n1);
473
474             $n1 = $self->{DOC_NOTES};
475             $n2 = $d->createElement('note');
476             $n2->setAttribute('xml:id', $self->{XMLID_PREFIX} . '_wc');
477             $n2->appendChild($d->createTextNode('Words: ' . $r->{'...
                    ...transcription_words_net'}));
478             $n1->appendChild($n2);
479             $n1->appendChild($self->{DOC_TRANSNOTES});
480             $self->{DOC_FILEDESC}->appendChild($n1);
481
482             $n1 = $self->{DOC_SOURCEDESC};
483             $n2 = $d->createElement('scriptStmt');
484             $n3 = $d->createElement('p');
485             $n3->appendChild($d->createTextNode('Latin Alphabet and ...
                    ...International Phonetic Alphabet'));
486             $n2->appendChild($n3);
487             $n1->appendChild($n2);
488             $n2 = $d->createElement('recordingStmt');
489             $n3 = $d->createElement('recording');
490             $n3->setAttribute('type','audio');
491             my $dur = $r->{'duration_of_recording'};
492             $dur =~ s/([0-9]{2,3}):([0-9]{2}):([0-9]{2})/PT${1}H${2}M${3}S/;
493             $n3->setAttribute('dur', $dur);
494             my $n4 = $d->createElement('date');
495             my $date = $r->{'date'};
496             $date =~ s/([0-9]{4})([0-9]{2})([0-9]{2})/$1-$2-$3/;
497             $n4->appendChild($d->createTextNode($date));
498             $n3->appendChild($n4);
499             $n4 = $d->createElement('equipment');
500             my $n5 = $d->createElement('p');
501             my $receq = $r->{'corp_h_equipment'}?$r->{'corp_h_equipment'}:'...
                    ...Portable minidisc recorder with electret condenser stereo ...
                    ...microphone';
502             $n5->appendChild($d->createTextNode($receq));
503             $n4->appendChild($n5);
504             $n3->appendChild($n4);
505             $n4 = $d->createElement('respStmt');
506             $n5 = $d->createElement('resp');
507             $n5->appendChild($d->createTextNode('recording'));
508             $n4->appendChild($n5);
509             {
510                 my $recorded_by = $r->{'corp_h_rec_pers'};
511                 foreach my $pers (split(/,/,$recorded_by)){
512                     $pers =~ s/[ \t]+/ /g;
513                     $n5 = $d->createElement('name');
514                     $n5->appendChild($d->createTextNode($pers));
515                     $n4->appendChild($n5);
516                 }
517             }
518             $n3->appendChild($n4);
519             $n2->appendChild($n3);
520             $n1->appendChild($n2);
521         }
522         $sh->finish();
523     }
524
525
526
527     # set the text classification (stratification)
528     {
```

```
529         my $textc = $d->createElement('textClass');
530         my $domc_l = '';
531         my $domc = '';
532         my $catr_l = $d->createElement('catRef');
533         my $catr = $d->createElement('catRef');
534         if($self->{DOMAIN}){
535             $domc .= '#' . $self->{DOMAIN} . ' ';
536             $domc_l .= '#' . $self->{XMLID_PREFIX} . '_' . $self->{DOMAIN} . '...
                ... ';
537         }
538         if($self->{SPET}){
539             $domc .= '#' . $self->{SPET} . ' ';
540             $domc_l .= '#' . $self->{XMLID_PREFIX} . '_' . $self->{SPET} . ' '...
                ...;
541         }
542 
543         if($domc || $domc_l){
544             $domc =~ s/ +$//;
545             $domc_l =~ s/ +$//;
546             $catr->setAttribute('target', $domc);
547             $catr_l->setAttribute('target', $domc_l);
548             $textc->appendChild($catr_l);
549             $textc->appendChild($catr);
550         }
551         $self->{DOC_PROFILEDESC}->appendChild($textc);
552     }
553 
554 
555 
556     #generate the settingDesc
557     {
558         my $settingDesc = $d->createElement('settingDesc');
559         my $setting = $d->createElement('setting');
560         if($self->{USE_DB} && $self->{DB_H}){
561             my $sh = $self->{DB_H}->prepare("SELECT corp_h_set_country AS ...
                ...country, corp_h_set_city AS city, corp_h_set_locale AS ...
                ...locale, corp_h_set_activity AS activity FROM events WHERE ...
                ...short_title = '" . $self->{H_SHORT_TITLE} . "'");
562             $sh->execute();
563             if(my $r = $sh->fetchrow_hashref()){
564                 if($r->{'country'}){
565                     my $n = $d->createElement('name');
566                     $n->setAttribute('type', 'country');
567                     $n->appendChild($d->createTextNode($r->{'country'}));
568                     $setting->appendChild($n);
569                 }
570                 if($r->{'city'}){
571                     my $n = $d->createElement('name');
572                     $n->setAttribute('type', 'city');
573                     $n->appendChild($d->createTextNode($r->{'city'}));
574                     $setting->appendChild($n);
575                 }
576                 if($r->{'locale'}){
577                     my $n = $d->createElement('locale');
578                     $n->appendChild($d->createTextNode($r->{'locale'}));
579                     $setting->appendChild($n);
580                 }
581                 if($r->{'activity'}){
582                     my $n = $d->createElement('activity');
583                     $n->appendChild($d->createTextNode($r->{'activity'}));
584                     $setting->appendChild($n);
585                 }
586             }
```

```
587             }
588             $settingDesc->appendChild($setting);
589             $self->{DOC_PROFILEDESC}->appendChild($settingDesc);
590         }
591
592
593         # generate the particDesc
594         {
595             $self->{DOC_PROFILEDESC}->appendChild($self->{DOC_PARTICDESC});
596             $self->{DOC_PARTICLIST} = $d->createElement('listPerson');
597             # workaround in case TEI does not change
598             $self->{DOC_PARTICLIST}->setAttribute('type','main');
599             $self->{DOC_PARTICDESC}->appendChild($self->{DOC_PARTICLIST});
600             my $list_ident = $d->createElement('listPerson');
601             $list_ident->setAttribute('type', 'identified');
602             my $list_unknown = $d->createElement('listPerson');
603             $list_unknown->setAttribute('type', 'not_identified');
604             my $list_groups =  $d->createElement('listPerson');
605             $list_groups->setAttribute('type', 'groups');
606
607             #groups of speakers
608             {
609                 my $sh = $self->{DB_H}->prepare("SELECT␣number_of_speakers,␣...
                        ...corp_h_num_interactants,␣corp_h_audience_size␣FROM␣events␣...
                        ...WHERE␣short_title␣=␣'" . $self->{H_SHORT_TITLE} . "'");
610                 $sh->execute();
611                 while(my $r = $sh->fetchrow_hashref()){
612                     if($r->{'number_of_speakers'}){
613                         my $grp = $d->createElement('personGrp');
614                         $grp->setAttribute('role', 'speakers');
615                         $grp->setAttribute('size', $r->{'number_of_speakers'});
616                         $list_groups->appendChild($grp);
617                     }
618                     if($r->{'corp_h_num_interactants'}){
619                         my $grp = $d->createElement('personGrp');
620                         $grp->setAttribute('role', 'interactants');
621                         $grp->setAttribute('size', $r->{'corp_h_num_interactants'...
                                ...});
622                         $list_groups->appendChild($grp);
623                     }
624                     if($r->{'corp_h_audience_size'}){
625                         my $grp = $d->createElement('personGrp');
626                         $grp->setAttribute('role', 'audience');
627                         $grp->setAttribute('size', $r->{'corp_h_audience_size'});
628                         $list_groups->appendChild($grp);
629                     }
630                 }
631                 $sh->finish();
632             }
633             $self->{DOC_PARTICLIST}->appendChild($list_groups);
634
635             if($self->{USE_DB} && $self->{DB_H}){
636                     my $stm = "SELECT␣speaker.*,␣speaker_event.*,␣"
637                 . "␣age_group.name␣AS␣age_group_name,␣"
638                 . "gender_codes.gender␣AS␣gender_desc␣"
639                 . "FROM␣speaker␣"
640                 . "LEFT␣JOIN(speaker_event,␣age_group,␣gender_codes)"
641                 . "␣ON␣(speaker.speaker_id␣=␣speaker_event.speaker_id␣"
642                 . "␣AND␣speaker.age␣=␣age_group.age_group_id␣"
643                 . "AND␣speaker.gender␣=␣gender_codes.code)␣"
644                 . "WHERE␣short_title␣=␣'" .
645                 $self->{H_SHORT_TITLE} . "'";
646 #                warn($stm);
```

```
647             my $sh = $self->{DB_H}->prepare($stm);
648                 $sh->execute();
649
650
651             while(my $r = $sh->fetchrow_hashref()){
652                 my $p = $d->createElement('person');
653                 my $role = $r->{'speaker_role'};
654                 $role =~ s/-//;
655                 $p->setAttribute('role', $r->{'speaker_role'});
656                 $p->setAttribute('sameAs', '#P' . $r->{'speaker_id'});
657                 if($r->{'speaker_id_transcript'}){
658                     $p->setAttribute('xml:id', $self->{XMLID_PREFIX} . '_' . ...
                            ...$r->{'speaker_id_transcript'});
659                     $p->setIdAttribute('xml:id');
660                     $self->{HAVE_SID_IN_PARTICDESC}->{$r->{'...
                            ...speaker_id_transcript'}} = 1;
661                 }
662                 else{
663                     warn "speaker's id not in database";
664                     $p->appendChild($d->createComment('speaker ' . $r->{'name'...
                            ...} . ' lacks a speaker_id in database')) if $self->{...
                            ...DEBUG};
665                     $p->setAttribute('voice:todo', 'SID in DB') if $self->{...
                            ...DEBUG};
666                 }
667
668                 if($r->{'name'}){
669                     my $pn = $d->createElement('persName');
670                     $pn->appendChild($d->createTextNode($r->{'name'}));
671                     $p->appendChild($pn);
672                 }
673
674                 {
675                         my $age = $d->createElement('age');
676                         $age->appendChild($d->createTextNode($r->{'...
                                ...age_group_name'}));
677                         $age->setAttribute('value', $r->{'age'});
678                         $p->appendChild($age);
679                 }
680
681                 {
682                         my $sex = $d->createElement('sex');
683                         $sex->appendChild($d->createTextNode($r->{'gender_desc...
                                ...'}));
684                         $sex->setAttribute('value', $r->{'gender'});
685                         $p->appendChild($sex);
686                 }
687
688
689
690                 if($r->{'occupation'}){
691                     my $po = $d->createElement('occupation');
692                     $po->appendChild($d->createTextNode($r->{'occupation'}));
693                     $p->appendChild($po);
694                 }
695                 if($r->{'info'}){
696                     my $note = $d->createElement('note');
697                     $note->appendChild($d->createTextNode($r->{'info'}));
698                     $p->appendChild($note);
699                 }
700
701                 my $lang_knowledge = $d->createElement('langKnowledge');
702                 $p->appendChild($lang_knowledge);
```

```
703                 for(my $i = 1; $r->{'L1_' . $i}; $i++){
704                     my $lk = $d->createElement('langKnown');
705                     $lk->setAttribute('level', 'L1');
706                     my $tag = $r->{'L1_' . $i};
707                     $tag .= '-' . $r->{'country_' . $i} if $r->{'country_' . ...
                            ...$i};
708                     $lk->setAttribute('tag', $tag);
709                     $lang_knowledge->appendChild($lk);
710                 }
711 
712                 $list_ident->appendChild($p);
713             }
714             $self->{DOC_PARTICLIST}->appendChild($list_ident);
715             $sh->finish();
716         }
717         else{
718             $self->{DOC_PARTICLIST}->appendChild($d->createComment('WARNING:␣...
                    ...not␣using␣database,␣falling␣back␣to␣simple␣mode')) if $self...
                    ...->{DEBUG};
719         }
720         foreach my $sid (keys(%{$self->{HAVE_SID_IN_PARTICDESC}})){
721             unless($self->{HAVE_SID_IN_PARTICDESC}->{$sid}){
722                 my $p = $d->createElement('person');
723                 $p->setAttribute('xml:id', $self->{XMLID_PREFIX} . '_' . $sid)...
                        ...;
724                 unless($sid =~ m/^SX/ || $sid =~ m/^SS/){
725                     $p->appendChild($d->createComment('WARNING:␣Speaker␣' . ...
                            ...$sid . '␣not␣in␣Database')) if $self->{DEBUG};
726                 }
727 
728                 if($sid =~ m/^SX-([1-9][0-9]*)/){
729                     $p->setAttribute('corresp', '#' . $self->{XMLID_PREFIX} . ...
                            ...'_S' . $1);
730                 }
731 
732                 if($sid =~ m/^(S(?:X(?:-f|-m)?|S))/){
733                         my $cur_sid = $1;
734                         $p->setAttribute('sameAs', '#' . $cur_sid);
735                 }
736 
737                 $list_unknown->appendChild($p);
738             }
739         }
740         if($list_unknown->getFirstChild()){
741             $self->{DOC_PARTICLIST}->appendChild($list_unknown);
742         }
743     }
744 
745         #speaker relations
746         {
747             if($self->{USE_DB} && $self->{DB_H}){
748                 my $sh = $self->{DB_H}->prepare("SELECT␣...
                        ...corp_h_num_interactants,␣corp_h_audience_size,␣␣...
                        ...number_of_speakers,␣power_relation,␣acquaintedness␣FROM␣...
                        ...events␣LEFT␣JOIN␣␣event_acquaintedness␣ON␣(events....
                        ...corp_h_acquaintedness=event_acquaintedness....
                        ...event_acquaintedness_id)␣LEFT␣JOIN␣event_power_relation␣...
                        ...ON␣(events.corp_h_power_rel=event_power_relation....
                        ...event_power_relation_id)␣WHERE␣events.short_title␣=␣'" ....
                        ... $self->{H_SHORT_TITLE} . "'");
749                 $sh->execute();
750                 my $rels = $d->createElement('relationGrp');
751                 my $linked = '';
```

```
752                 for my $sid (sort(keys(%{$self->{HAVE_SID_IN_PARTICDESC}}))){
753                     my $ignore_me = $self->{DOC}->getElementById($self->{...
                            ...XMLID_PREFIX} . '_' . $sid);
754                     if($ignore_me){
755                         if($ignore_me->getAttribute('role') eq 'researcher' ||
756                            $ignore_me->getAttribute('role') eq 'non-...
                                  ...participant'){
757                             $ignore_me = 1;
758                         }
759                         else{
760                             $ignore_me = 0;
761                         }
762                     }
763                     unless($sid =~ m/^SX/ || $sid =~ m/^SS/ || $ignore_me){
764                         $linked .= '#' . $self->{XMLID_PREFIX} . '_' . $sid . ...
                                ...' ';
765                     }
766                 }
767                 $linked =~ s/ +$//;
768                 while(my $r = $sh->fetchrow_hashref()){
769                     if($r->{'power_relation'}){
770                         my $rel = $d->createElement('relation');
771                         $rel->setAttribute('name', $self->normalize_space($r...
                                ...->{'power_relation'}));
772                         $rel->setAttribute('type', 'power');
773                         $rel->setAttribute('mutual', $linked);
774                         $rels->appendChild($rel);
775                     }
776                     if($r->{'acquaintedness'}){
777                         my $rel = $d->createElement('relation');
778                         $rel->setAttribute('name', $self->normalize_space($r...
                                ...->{'acquaintedness'}));
779                         $rel->setAttribute('type', 'acquaintedness');
780                         $rel->setAttribute('mutual', $linked);
781                         $rels->appendChild($rel);
782                     }
783 
784                 }
785                 $self->{DOC_PARTICLIST}->appendChild($rels);
786 
787                 $sh->finish();
788             }
789         }
790 
791 
792     $self->{DOC_FILEDESC}->appendChild($self->{DOC_NOTES});
793     $self->{DOC_FILEDESC}->appendChild($self->{DOC_SOURCEDESC});
794     my $n = $d->createElement('change');
795     $n->appendChild($d->createTextNode('initial conversion M1'));
796     $self->{DOC_REVISIONDESC}->appendChild($self->generate_revision_desc());
797 }
798 
799 #
800 # generate the revisionDesc for the text header
801 #
802 sub generate_revision_desc(){
803     my ($self) = @_;
804     if($self->{USE_DB}){
805         unless($self->{DB_H}){
806             $self->connect_db();
807         }
808     }
809     my $d = $self->{DOC};
```

```
810     my $df = $d->createDocumentFragment();
811     my $last_inserted = '';
812 
813     foreach my $rev (
814                         sort {
815                             my $left = $self->{REVISION}->{$VoiceParser::a}->{...
                                ...revdate};
816                             $left =~ s/-//g;
817                             my $right = $self->{REVISION}->{$VoiceParser::b}->{...
                                ...revdate};
818                             $right =~ s/-//g;
819                             $left <=> $right;
820                         } keys(%{$self->{REVISION}})){
821         my $revtype = $self->{REVISION}->{$rev}->{revtype};
822         foreach my $who ($self->{REVISION}->{$rev}->{who}){
823                         my $n = $d->createElement('change');
824                         $n->setAttribute('who', $self->{REVISION}->{$rev}->{...
                            ...who});
825                         $n->setAttribute('when', $self->{REVISION}->{$rev}->{...
                            ...revdate});
826                         $n->appendChild($d->createTextNode($revtype));
827 
828                         if(!$n->hasChildNodes()){
829                                 $df->appendChild($n);
830                                 $last_inserted = $n;
831                         }
832                         else {
833                                 $df->insertBefore($n, $df->getFirstChild());
834                                 $last_inserted = $n;
835                         }
836                 }
837     }
838 
839     my $time = Date::ISO->new( epoch => time());
840     my $n = $d->createElement('change');
841     $n->setAttribute('who', 'SM');
842     $n->setAttribute('when', $time->year . '-' . ($time->month<10?'0':'') . ...
        ...$time->month . '-' . ($time->day<10?'0':'') . $time->day);
843     $n->appendChild($d->createTextNode('conversion to XML'));
844     $df->insertBefore($n, $df->getFirstChild());
845     return $df;
846 }
847 
848 #
849 # generate the encodingDesc for the text-header
850 #
851 sub generate_encoding_desc(){
852     my ($self) = @_;
853     if($self->{USE_DB}){
854         unless($self->{DB_H}){
855             $self->connect_db();
856         }
857     }
858     my $d = $self->{DOC};
859     my $classd = $d->createElement('classDecl');
860     my $tax = $d->createElement('taxonomy');
861     $classd->appendChild($tax);
862     my $spets = $d->createElement('category');
863     $spets->setAttribute('xml:id', $self->{XMLID_PREFIX} . '_' . 'spet');
864     my $catd = $d->createElement('catDesc');
865     $catd->appendChild($d->createTextNode('Speech Event Type'));
866     $spets->appendChild($catd);
867     my $doms = $d->createElement('category');
```

```
868     $doms->setAttribute('xml:id', $self->{XMLID_PREFIX} . '_' . 'domain');
869     $catd = $d->createElement('catDesc');
870     $catd->appendChild($d->createTextNode('Domain'));
871     $doms->appendChild($catd);
872 
873     $tax->appendChild($doms);
874     $tax->appendChild($spets);
875 
876     if($self->{USE_DB} && $self->{DB_H}){
877         {
878             my $q = 'SELECT * FROM corpus_domains ORDER BY domain_id';
879             my $sh = $self->{DB_H}->prepare($q);
880             $sh->execute();
881             while(my $r = $sh->fetchrow_hashref()){
882                 my $cat = $d->createElement('category');
883                 $cat->setAttribute('xml:id', $self->{XMLID_PREFIX} . '_' . $r...
                        ...->{'xml_id_comp'});
884                 $cat->setAttribute('sameAs', '#' . $r->{'xml_id_comp'});
885                 $catd = $d->createElement('catDesc');
886                 $catd->appendChild($d->createTextNode($r->{'domain'}));
887                 $cat->appendChild($catd);
888                 $doms->appendChild($cat);
889             }
890         }
891         {
892             my $q = 'SELECT * FROM corpus_spets ORDER BY spet_id';
893             my $sh = $self->{DB_H}->prepare($q);
894             $sh->execute();
895             while(my $r = $sh->fetchrow_hashref()){
896                 my $cat = $d->createElement('category');
897                 $cat->setAttribute('xml:id', $self->{XMLID_PREFIX} . '_' . $r...
                        ...->{'xml_id_comp'});
898                 $cat->setAttribute('sameAs', '#' . $r->{'xml_id_comp'});
899                 $catd = $d->createElement('catDesc');
900                 $catd->appendChild($d->createTextNode($r->{'spet'}));
901                 $cat->appendChild($catd);
902                 $spets->appendChild($cat);
903             }
904         }
905     }
906     $self->{DOC_ENCODINGDESC}->appendChild($classd);
907     $self->{DOC_ENCODINGDESC}->appendChild($d->createElement('tagsDecl'));
908 }
909 
910 
911 
912 
913 
914 #### several groupings of parser methods
915 
916 #
917 # parse: sequence of word parts
918 #
919 sub parse_wp_stretch(\$$){
920     my ($self, $line, $d) = @_;
921     if(ref($line) eq 'REF'){
922         $line = $$line;
923     }
924     my $node = '';
925     if($node = $self->parse_space(\$line, $d)){}
926     elsif($node = $self->parse_wp(\$line, $d)){}
927     elsif($node = $self->parse_pause(\$line, $d)){}
928     elsif($node = $self->parse_length(\$line, $d)){}
```

```
929     elsif($node = $self->parse_uncertain(\$line, $d)){}
930     elsif($node = $self->parse_unintell(\$line, $d)){}
931     elsif($node = $self->parse_contextual(\$line, $d)){}
932     elsif($node = $self->parse_rise(\$line, $d)){}
933     elsif($node = $self->parse_fall(\$line, $d)){}
934     elsif($node = $self->parse_lx(\$line, $d)){}
935     elsif($node = $self->parse_sm_laugh(\$line, $d)){}
936     elsif($node = $self->parse_hash(\$line, $d)){}
937     elsif($node = $self->parse_excl(\$line, $d)){}
938     elsif($node = $self->parse_directed(\$line, $d)){}
939     elsif($node = $self->parse_track(\$line, $d)){}
940     elsif($node = $self->parse_ol(\$line, $d)){}
941     elsif($node = $self->parse_othercont(\$line, $d)){}
942     elsif($node = $self->parse_ono(\$line, $d)){}
943     elsif($node = $self->parse_excl_hash_voice(\$line, $d)){}
944     #### open list items!!! come last, but handle with care
945     elsif($node = $self->parse_sm(\$line, $d)){}
946     elsif($node = $self->parse_sn(\$line, $d)){}
947     return $node;
948     }
949
950 #
951 # parse: not counting as word part
952 #
953 sub parse_wp_nw(\$$){
954     my($self, $line, $d) = @_;
955     if(ref($line) eq 'REF'){
956         $line = $$line;
957     }
958     my $n = '';
959     if($n = $self->parse_wpl(\$line, $d)){}
960     elsif($n = $self->parse_wpu(\$line, $d)){}
961     elsif($n = $self->parse_wplaugh(\$line, $d)){}
962     elsif($n = $self->parse_wph(\$line, $d)){}
963     elsif($n = $self->parse_anon(\$line, $d)){}
964     elsif($n = $self->parse_spel(\$line, $d)){}
965     elsif($n = $self->parse_pvc(\$line, $d)){}
966     elsif($n = $self->parse_wpyear(\$line, $d)){}
967
968     return $n;
969 }
970
971 #
972 # parse: count as word part
973 #
974 sub parse_wp(\$$){
975     my($self, $line, $d) = @_;
976     if(ref($line) eq 'REF'){
977         $line = $$line;
978     }
979     my $n = '';
980     if($n = $self->parse_wp_nw(\$line, $d)){}
981     if($n){
982         if($self->{FINISHED_UTTERANCE}){
983             die('LINE: ' . $self->{LINE_CNT} . ' parsed word-part after other ...
                    ...continuation=');
984         }
985         $self->{FRESH_UTTERANCE} = 0;
986     }
987     return $n;
988
989 }
990
```

```
991
992  ### helper functions for actual text parsing
993
994  #
995  # helper method: removes meaningless linebreaks between
996  # header->body->transcriber_notes
997  #
998  sub eat_nl(\$){
999      my ($self, $line) = @_;
1000     if(ref($line) eq 'REF'){
1001         $line = $$line;
1002     }
1003     if($$line =~ s/^\r?\n//){
1004 #       print STDERR 'Line: ' . $self->{LINE_CNT} . "\n";
1005         return $self->{LINE_CNT}++;
1006     }
1007     return '';
1008 }
1009
1010 #
1011 # helper method: determines the type of revision information in header
1012 #
1013 sub get_revtype($){
1014     my($self, $revkey) = @_;
1015     $revkey = lc($revkey);
1016     my $revtype = $revkey;
1017     if($revkey eq 'transcription'){$revtype = 'transcription';}
1018     elsif($revkey eq 'cv'){$revtype = 'transcript conversion';}
1019     elsif($revkey eq 'c0'){$revtype = 'transcript completion';}
1020     elsif($revkey eq 'c1'){$revtype = 'checking';}
1021     elsif($revkey eq 'c2'){$revtype = 'proof-reading';}
1022     elsif($revkey =~ m/^p[1-9]\.[0-9]/){$revtype = 'parsing';}
1023     elsif($revkey =~ m/^v[1]$/){$revtype = 'finalized for publication';}
1024 }
1025
1026 #
1027 # helper method: adds a revision to the parse-tree
1028 #
1029 sub add_revision($$){
1030     my($self, $revkey, $who) = @_;
1031     my $revtype = $self->get_revtype($revkey);
1032     $revkey = lc($revkey);
1033     $who =~ s/,[ \t]*/ /g;
1034     unless($self->{REVISION}->{$revkey}){
1035         $self->{REVISION}->{$revkey} = {who => $who,
1036                                         revtype => $revtype,
1037                                         revdate => ''
1038                                         };
1039     }
1040     else{
1041                 $self->{REVISION}->{$revkey}->{who} .= " " . $who;
1042 #               warn "Second revision entry for " . $revkey . ": " . $who;
1043     }
1044 }
1045
1046 #
1047 # helper method: sets the date for a revision
1048 #
1049 sub set_revdate($){
1050     my($self, $revkey, $revdate) = @_;
1051     $revkey = lc($revkey);
1052     # only allowed after by line?
1053     unless($self->{REVISION}->{$revkey}){
```

```
1054            die "no revision for key: " . $revkey . " date: " . $revdate;
1055        }
1056        if($revdate =~ m/([0-9]{4})([0-9]{2})([0-9]{2})/){
1057            my $year = $1;
1058            my $mon = $2;
1059            my $day = $3;
1060            my $rev = $year . "-" . $mon;
1061            if($day ne "00"){
1062                $rev .= "-" . $day;
1063            }
1064            $self->{REVISION}->{$revkey}->{revdate} = $rev;
1065        }
1066    }
1067
1068    ########################################################################
1069    #  the individual parsers
1070    ########################################################################
1071
1072    #
1073    # parser: the transcript --> starting point for parsing the text
1074    #
1075    sub parse_transcript (\$$){
1076        my ($self, $line, $d) = @_;
1077        if(ref($line) eq 'REF'){
1078            $line = $$line;
1079        }
1080        unless($d){
1081            $d = $self->{DOC};
1082        }
1083        $self->parse_header(\$line, $d);
1084        $self->{DOC_BODY}->appendChild($self->parse_body(\$line, $d));
1085        if(my $n = $self->parse_transcriber_notes(\$line, $d)){
1086            $self->{DOC_TRANSNOTES}->appendChild($n);
1087        }
1088        else{
1089            die "failed to parse transcriber notes!!!!";
1090        }
1091        $self->eat_nl();
1092    }
1093
1094    #
1095    # parser: declaration of transcription type
1096    #
1097    sub parse_VOICE_token (\$$){
1098        my ($self, $line, $d) = @_;
1099        if(ref($line) eq 'REF'){
1100            $line = $$line;
1101        }
1102        if($$line =~ s/VOICE[ \t]*//){
1103            $self->eat_nl($line);
1104            return 1;
1105        }
1106        return '';
1107    }
1108
1109    #
1110    # parser: parse the header of the transcript.
1111    #
1112    sub parse_header (\$$){
1113        my ($self, $line, $d) = @_;
1114        if(ref($line) eq 'REF'){
1115            $line = $$line;
1116        }
```

```
1117
1118      my $pat_char = '[a-zA-Z0-9]+';
1119      my $pat_l1s = '[a-zA-Z0-9,␣()-/?]+';
1120      my $pat_date = '[0-9]{8}';
1121      my $pat_time = '[0-9x]{2,3}:[0-9x]{2}:[0-9x]{2}';
1122      my $pat_num = '[0-9]+';
1123      $self->parse_VOICE_token(\$line, $d) or die('Transcript␣does␣not␣start␣...
              ...with␣"VOICE":␣' . substr($$line, 0, 50));
1124      while(1){
1125          if($$line =~ s/^(Short title):[ \t]+($pat_char)[ \t]*//){
1126              $self->{H_SHORT_TITLE} = $2;
1127              if($self->{USE_DB}){
1128                  $self->set_id_prefix_db();
1129              }
1130          } # FIXME pattern for multiple resps
1131          elsif($$line =~ s/^(Date of event):[ \t]+($pat_date)[ \t]*//){}
1132          elsif($$line =~ s/^(Duration of whole recording):[ \t]+($pat_time)[ \t...
                 ...]*//){}
1133          elsif($$line =~ s/^(Transcript duration):[ \t]+($pat_time)[ \t]*//){}
1134          elsif($$line =~ s/^(Number of words transcribed):[ \t]+($pat_num)[ \t...
                 ...]*//){}
1135          # ignore parser runs
1136          elsif($$line =~ s/^(P[1-9]\.[0-9]) by:[ \t]+($pat_char)(?:,[ \t]*...
                 ...$pat_char)*[ \t]*//){}
1137          elsif($$line =~ s/^(Transcription|CV|C0|C1|C2|V[1]) by:[ \t]+((?:...
                 ...$pat_char)(?:,[ \t]*$pat_char)*)[ \t]*//){
1138              $self->add_revision($1, $2);
1139          }
1140          elsif($$line =~ s/^Time required for (transcription|CV|C0|C1|C2):[ \t...
                 ...]+($pat_time)[ \t]*//){}
1141          # ignore parser runs
1142          elsif($$line =~ s/^(?:Last )?(P[1-9]\.[0-9]) date:[ \t]+($pat_date)[ \...
                 ...t]*//){}
1143          elsif($$line =~ s/^(?:Last )?(transcription|CV|C0|C1|C2|V[1]|P...
                 ...[1-9]\.[0-9]) date:[ \t]+($pat_date)[ \t]*//){
1144              $self->set_revdate($1,$2);
1145          }
1146
1147          elsif($$line =~ s/^(Number of speakers):[ \t]+($pat_num)[ \t]*//){}
1148          elsif($$line =~ s/^(L1s):[ \t]+($pat_l1s)[ \t]*//){}
1149
1150          elsif($$line =~ s/^(Overlap distance):[ \t]+($pat_num)[ \t]*//){
1151              $self->set_disambiguation_depth($2);
1152          }
1153          elsif($self->eat_nl($line)){}
1154          else{
1155              print STDERR 'finished␣reading␣transcript-header' . "\n";
1156              last;
1157          }
1158
1159      }
1160  }
1161
1162  #
1163  # parser: gaps in transcription
1164  #
1165  sub parse_gap(\$$){
1166      my ($self, $line, $d) = @_;
1167      if(ref($line) eq 'REF'){
1168          $line = $$line;
1169      }
1170      if($$line =~ s/^\(gap[ \t]+(([0-9]{2,3}):([0-9]{2}):([0-9]{2}))\)[ \t...
             ...]*\{(.*?)\}[ \t]*//s){
```

```
1171 #       warn "in gap!!!\n";
1172         my ($time, $time_h, $time_m, $time_s, $desc) = ($1, $2, $3, $4, $5);
1173         $time = $time_h?$time_h:0;
1174         my $g = $d->createElement('gap');
1175         $g->setAttribute('reason', 'not_transcribed');
1176         $g->setAttribute('dur', 'PT' . $time_h . 'H' . $time_m . 'M' . $time_s...
                 ... . 'S');
1177         $g->setAttribute('voice:desc', $desc);
1178 #       warn "endof gap!!!\n";
1179         return $g;
1180     }
1181     return '';
1182 }
1183
1184 #
1185 # parser: gaps in recording
1186 #
1187 sub parse_nrec(\$$){
1188     my ($self, $line, $d) = @_;
1189     if(ref($line) eq 'REF'){
1190         $line = $$line;
1191     }
1192     if($$line =~ s/^\(nrec[ \t]+(([0-9]{2,3}):([0-9]{2}):([0-9]{2}))\)[ \t...
            ...]*\{(.*?)\}[ \t]*//s){
1193         my ($time, $time_h, $time_m, $time_s, $desc) = ($1, $2, $3, $4, $5);
1194         my $g = $d->createElement('gap');
1195         $g->setAttribute('reason', 'not_recorded');
1196         $g->setAttribute('dur', 'PT' . $time_h . 'H' . $time_m . 'M' . $time_s...
                 ... . 'S');
1197         $g->setAttribute('voice:desc', $desc);
1198         return $g;
1199     }
1200     return '';
1201 }
1202
1203 #
1204 # parser: block of transcription related to a source medium
1205 #
1206 sub parse_medium(\$$){
1207     my ($self, $line, $d) = @_;
1208     if(ref($line) eq 'REF'){
1209         $line = $$line;
1210     }
1211     if($$line =~ s/^<beg[ \t]+([A-Za-z]{2,3})([0-9]{1,2}(?:\([a-z]\))?)_...
            ...([0-9]{1,2})_((?:[0-9]{2}:)?[0-9]{2}:[0-9]{2})[ \t]*>[ \t]*(.*?)<end...
            ...[ \t]+([A-Za-z]{2,3})([0-9]{1,2}(?:\([a-z]\))?)_([0-9]{1,2})_...
            ...((?:[0-9]{2}:)?[0-9]{2}:[0-9]{2})[ \t]*>[ \t]*//s){
1212         my $subline = $5;
1213         my @beg = ($1, $2, $3, $4);
1214         my @end = ($6, $7, $8, $9);
1215         my $med = $d->createElement('div');
1216         $med->setAttribute('type', 'medium');
1217         $med->setAttribute('voice:start', $beg[3]);
1218         $med->setAttribute('voice:end', $end[3]);
1219         $med->setAttribute('voice:start_track', $beg[2]);
1220         $med->setAttribute('voice:end_track', $end[2]);
1221         $med->setAttribute('voice:medium_type', ($end[0] eq $beg[0]?$end[0]:...
                ...$beg[0] . '/' . $end[0]));
1222         $med->setAttribute('voice:start_medium_number', $beg[1]);
1223         $med->setAttribute('voice:end_medium_number', $end[1]);
1224         while(length($subline) > 0){
1225             my $n = '';
1226             if($n = $self->parse_track(\$subline, $d)){}
```

```
1227            elsif($self->eat_nl(\$subline)){}
1228            elsif($n = $self->parse_hash(\$subline, $d)){}
1229            elsif($n = $self->parse_excl(\$subline, $d)){}
1230            elsif($n = $self->parse_u(\$subline, $d)){}
1231            elsif($subline =~ s/^[ \t]+//){}
1232            else{
1233                die ('LINE: ' . $self->{LINE_CNT} . "failed in medium: " .
1234                    substr($subline, 0, 50));
1235            }
1236            if($n){
1237                $med->appendChild($n);
1238            }
1239        }
1240        return $med;
1241    }
1242    return '';
1243 }
1244
1245 #
1246 # parser: body of the transcript
1247 #
1248 # does not return false on failure
1249 sub parse_body(\$$){
1250     my ($self, $line, $d) = @_;
1251     if(ref($line) eq 'REF'){
1252         $line = $$line;
1253     }
1254     my $b = $d->createDocumentFragment();
1255     while(1){
1256         my $n = '';
1257         if($n = $self->parse_medium($line, $d)){}
1258         elsif($n = $self->parse_gap($line, $d)){}
1259         elsif($n = $self->parse_nrec($line, $d)){}
1260         elsif($n = $self->parse_hash($line, $d)){}
1261         elsif($n = $self->parse_excl($line, $d)){}
1262         elsif($self->eat_nl($line)){}
1263         else{
1264             $self->status('finished body at: ' . substr($$line, 0, 50));
1265             last;
1266         }
1267         if($n){
1268             $b->appendChild($n);
1269         }
1270     }
1271     return $b;
1272 }
1273
1274 #
1275 # parser: text in transcriber notes
1276 #
1277 sub parse_tn_text(\$$){
1278     my ($self, $line, $d) = @_;
1279     if(ref($line) eq 'REF'){
1280         $line = $$line;
1281     }
1282     #FIXME pattern
1283     if($$line =~ s/^(.*?)(?=((?:<((?:#|!)[0-9]+)>.*?<\/\3>)|(?:<\/transcriber ...
         ...notes>)|$))//s){
1284         my $t = $1;
1285         $t =~ s/(?:\r|\n)+/ /g;
1286         return $d->createTextNode($t);
1287     }
1288     return '';
```

```
1289 }
1290
1291 #
1292 # parser: block of transcriber notes
1293 #
1294 sub parse_transcriber_notes(\$$){
1295     my ($self, $line, $d) = @_;
1296     if(ref($line) eq 'REF'){
1297         $line = $$line;
1298     }
1299     if($$line =~ s/^<transcriber notes>(.*?)<\/transcriber notes>//s){
1300         print STDERR "parsing␣transcriber␣notes\n";
1301         my $subline = $1;
1302         my $tn = $d->createDocumentFragment();
1303         my $general_cnt = 0;
1304         while(length($subline) > 0){
1305 #           warn "SUBLINE " . substr($subline, 0, 50) . "\n";
1306             my $n = '';
1307             if($subline =~ s/^[ \t]+//){}
1308             elsif($self->eat_nl(\$subline)){}
1309             elsif($n = $self->parse_tn_hash(\$subline, $d)){}
1310             elsif($n = $self->parse_tn_excl(\$subline, $d)){}
1311 #           elsif($n = $self->parse_tn_prose(\$subline, $d)){}
1312             elsif($n = $self->parse_tn_text(\$subline, $d)){
1313                 my $n1 = $d->createElement('note');
1314                 $n1->setAttribute('xml:id', $self->{XMLID_PREFIX} . '_tn_' . ...
                        ...$general_cnt++);
1315                 $n1->appendChild($n);
1316                 $n = $n1;
1317             }
1318             else{
1319                 die "LINE␣" . $self->{LINE_CNT} . ":␣Failed␣in␣<transcriber␣...
                        ...notes>:␣"
1320                     . substr($subline, 0, 50);
1321             }
1322             if($n){
1323
1324                 $tn->appendChild($n);
1325             }
1326         }
1327         return $tn;
1328     }
1329     return '';
1330 }
1331
1332 #
1333 # parse: utterance -> speaker contribution
1334 #
1335 sub parse_u (\$$){
1336     my ($self, $line, $d) = @_;
1337     if(ref($line) eq 'REF'){
1338         $line = $$line;
1339     }
1340
1341     my $cn = $d->createElement('u');
1342
1343     unless($self->parse_sid(\$line, $cn)){
1344         return '';
1345     }
1346     #strip whitespace at start and end;
1347     $$line =~ s/^[ \t]+//;
1348     $$line =~ s/[ \t]+$//;
1349     #strip weirdnesses
```

```
1350      $$line =~ s/\342\200\231/\047/g;
1351      if($$line =~ s/^(.+?\r?\n)//m){
1352          my $subline = $1;
1353          $self->{FRESH_UTTERANCE} = 1;
1354          $self->{FINISHED_UTTERANCE} = 0;
1355          $self->{OVERLAP_IN_UTTERANCE} = 0;
1356          $self->{NO_WORD} = 1;
1357
1358          $cn->setAttribute('xml:id', $self->{XMLID_PREFIX} . '_u_' . ++$self->{...
                ...UTTERANCE_CNT});
1359
1360          while(length($subline) > 0){
1361              my $node = 0;
1362              if($node = $self->parse_wp_stretch(\$subline, $d)){
1363                      # check for space!
1364                  }
1365              elsif($self->eat_nl(\$subline)){}
1366              else{
1367                  die ('LINE: ' . $self->{LINE_CNT} . " Failed to parse in ...
                        ...utterance at: '"
1368                   . substr($subline, 0, 50) . "'\n\nreturning anyway\n");
1369
1370              }
1371              if($node){
1372                  $cn->appendChild($node);
1373              }
1374          }
1375                  # add a final space
1376                  $cn->appendChild($d->createTextNode(' '));
1377          if(!$self->{OVERLAP_IN_UTTERANCE}){
1378              foreach my $ol (keys(%{$self->{OVERLAP_REFNUM}})){
1379                  die ('LINE: ' . $self->{LINE_CNT} . " dangling overlap $ol: "...
                        ... . substr($$line, 0, 50)) if $self->{OVERLAP_REFNUM}->{...
                        ...$ol} == 0;
1380                  $self->status("more than two references to overlap ${ol}! is ...
                        ...that sane?") if $self->{OVERLAP_REFNUM}->{$ol} > 2;
1381              }
1382              $self->{OVERLAP_MAP} = {};
1383              $self->{OVERLAP_REFNUM} = {};
1384          }
1385          return $cn;
1386      }
1387      return '';
1388  }
1389
1390  #
1391  # parser: sigle referencing speaker
1392  #
1393  sub parse_sid (\$$){
1394      my ($self, $line, $n) = @_;
1395      if(ref($line) eq 'REF'){
1396          $line = $$line;
1397      }
1398      if($$line =~ s/^(S([1-9][0-9]?|(X(-(?:[mf]|[1-9][0-9]?))?)|S)):[ \t]*//){
1399          my $who = $1;
1400          $n->setAttribute('who', '#'.$self->{XMLID_PREFIX}.'_'.$who);
1401          $self->{HAVE_SID_IN_PARTICDESC}->{$who} = '';
1402          return 1;
1403      }
1404      return '';
1405  }
1406
1407  #
```

```
1408 # parser: space in text, reset word status
1409 #
1410 sub parse_space (\$$){
1411     my ($self, $line, $d) = @_;
1412     if(ref($line) eq 'REF'){
1413         $line = $$line;
1414     }
1415     if($$line =~ s/^[ \t]+//){
1416                 if(!$self->{NO_WORD}){
1417                         $self->{NO_WORD} = 1;
1418                         return $d->createTextNode('␣');
1419                 }
1420                 else{
1421                         return $d->createTextNode('');
1422                 }
1423     }
1424     return '';
1425 }
1426
1427 #
1428 # parser: parse uppercase word-part --> emphasis
1429 #
1430 sub parse_wpu (\$$){
1431     my ($self, $line, $d) = @_;
1432     if(ref($line) eq 'REF'){
1433         $line = $$line;
1434     }
1435     if($$line =~ s/^([A-Z\']+)//){␣#'
1436         my $wpu = lc($1);
1437         my $seg = $d->createElement('emph');
1438         $self->{NO_WORD} = 0;
1439         $seg->appendChild($d->createTextNode($wpu));
1440         return $seg;
1441     }
1442     return '';
1443 }
1444
1445 #
1446 # parser: parse lower-case word-part
1447 #
1448 sub parse_wpl (\$$){
1449     my ($self, $line, $d) = @_;
1450     if(ref($line) eq 'REF'){
1451         $line = $$line;
1452     }
1453     if($$line =~ s/^([a-z\']+)//){␣#'
1454         my $str = $1;
1455         $self->{NO_WORD} = 0;
1456
1457         return $d->createTextNode($str);
1458     }
1459     return '';
1460 }
1461
1462 #
1463 # parser: hyphens indicating incomplete word
1464 #
1465 sub parse_wph (\$$){
1466     my ($self, $line, $d) = @_;
1467     if(ref($line) eq 'REF'){
1468         $line = $$line;
1469     }
1470     my $str = '-';
```

```
1471      if($$line =~ s/^-//){
1472          return $d->createTextNode($str);
1473      }
1474      return '';
1475  }
1476
1477  #
1478  # parser: parse laughter
1479  #
1480  sub parse_wplaugh (\$$){
1481      my ($self, $line, $d) = @_;
1482      if(ref($line) eq 'REF'){
1483          $line = $$line;
1484      }
1485      if($$line =~ s/^([@]+)//){
1486          my $syl = length($1);
1487          my $v = $d->createElement('vocal');
1488          $v->setAttribute('type', 'wordlike');
1489          if ($self->{NO_WORD} == 1){
1490              $v->setAttribute('subtype', 'wordstart');
1491              $self->{NO_WORD} = 0;
1492          }
1493          $v->setAttribute('voice:syl', $syl);
1494          $v->setAttribute('voice:desc', 'laughing');
1495          return $v;
1496      }
1497      return '';
1498  }
1499
1500  #
1501  # parser: a numeric indicating a year
1502  #
1503  sub parse_wpyear (\$$){
1504      my ($self, $line, $d) = @_;
1505      if(ref($line) eq 'REF'){
1506          $line = $$line;
1507      }
1508      if($$line =~ s/^([0-9]{3,})//){
1509          my $str = $1;
1510          $self->{NO_WORD} = 0;
1511          return $d->createTextNode($str);
1512      }
1513      return '';
1514  }
1515
1516  #
1517  # parser: a pause in speech
1518  #
1519  sub parse_pause (\$$){
1520      my ($self, $line, $d) = @_;
1521      if(ref($line) eq 'REF'){
1522          $line = $$line;
1523      }
1524      if($$line =~ s/^\((((\.)|([1-9]?[0-9]*))\)//){
1525          my $e = $d->createElement('pause');
1526          $e->setAttribute('dur', 'PT' . $3 . 'S') if $3;
1527          $self->{NO_WORD} = 1;
1528          return $e;
1529      }
1530      return '';
1531  }
1532
1533
```

```
1534 #
1535 # parser: lengthening in/at end of word
1536 #
1537
1538 # TEI Workshop. As our transcription conventions use the colon, we can
1539 # stick to this.
1540
1541 sub parse_length (\$$){
1542     my ($self, $line, $d) = @_;
1543     if(ref($line) eq 'REF'){
1544         $line = $$line;
1545     }
1546     if(!$self->{NO_WORD} && $$line =~ s/^://){
1547         my $n = $d->createElement('c');
1548         $n->setAttribute('type', 'lengthening');
1549         return $n;
1550     }
1551     return '';
1552 }
1553
1554 #
1555 # parser: rising intonation
1556 #
1557 sub parse_rise (\$$){
1558     my ($self, $line, $d) = @_;
1559     if(ref($line) eq 'REF'){
1560         $line = $$line;
1561     }
1562     if(!$self->{NO_WORD} && $$line =~ s/^\?//){
1563         my $n = $d->createElement('c');
1564         $n->setAttribute('type', 'intonation');
1565         $n->setAttribute('function', 'rise');
1566         return $n;
1567     }
1568     return '';
1569 }
1570 sub parse_fall (\$$){
1571     my ($self, $line, $d) = @_;
1572     if(ref($line) eq 'REF'){
1573         $line = $$line;
1574     }
1575     if(!$self->{NO_WORD} && $$line =~ s/^\.//){
1576         my $n = $d->createElement('c');
1577         $n->setAttribute('type', 'intonation');
1578         $n->setAttribute('function', 'fall');
1579         return $n;
1580     }
1581     return '';
1582 }
1583
1584 #
1585 # parser: uncertain speech
1586 #
1587 sub parse_uncertain (\$$){
1588     my ($self, $line, $d) = @_;
1589     if(ref($line) eq 'REF'){
1590         $line = $$line;
1591     }
1592     if($$line =~ s/^\((([^()]|\((?:\.|[1-9][0-9]?)\))+?)\)//){
1593         my $subline =  $1;
1594         my $uc = $d->createElement('unclear');
1595         while(length($subline)>0){
1596             my $n = 0;
```

```
1597                if($n = $self->parse_wp_stretch(\$subline, $d)){}
1598                else{
1599                    die ('LINE: ' . $self->{LINE_CNT}
1600                 . 'failed to parse in (uncertain) at: ' . substr($$line, 0, 50));
1601                }
1602                if($n){
1603                    $uc->appendChild($n);
1604                }
1605            }
1606            return $uc;
1607        }
1608        return '';
1609    }
1610
1611    #
1612    # parser: unintelligible speech
1613    #
1614    sub parse_xl(\$$){
1615        my ($self, $line, $d) = @_;
1616        if(ref($line) eq 'REF'){
1617            $line = $$line;
1618        }
1619        if($$line =~ s/^(x+)//){
1620            my $str = $1;
1621            $self->{NO_WORD} = 0;
1622            return $d->createTextNode($str);
1623        }
1624        return '';
1625    }
1626
1627    #
1628    # parser: unintelligible, emphasised speech
1629    #
1630    sub parse_xu(\$$){
1631        my ($self, $line, $d) = @_;
1632        if(ref($line) eq 'REF'){
1633            $line = $$line;
1634        }
1635        if($$line =~ s/^(X+)//){
1636            my $subline = $1;
1637            $self->{NO_WORD} = 0;
1638            return $self->parse_wpu(\$subline, $d);
1639        }
1640        return '';
1641    }
1642
1643    #
1644    # parser: unintelligible speech
1645    #
1646    sub parse_unintell (\$$){
1647        my ($self, $line, $d) = @_;
1648        if(ref($line) eq 'REF'){
1649            $line = $$line;
1650        }
1651        if($$line =~ s/^<un>(.+?)<\/un>//){
1652            my $subline = $1;
1653            my $ui = $d->createElement('supplied');
1654            $ui->setAttribute('reason', 'unintelligible');
1655            while(length($subline)>0){
1656                my $n = 0;
1657                if($n = $self->parse_space(\$subline, $d)){}
1658                elsif($n = $self->parse_xl(\$subline, $d)){}
1659                elsif($n = $self->parse_xu(\$subline, $d)){}
```

```
1660                elsif($n = $self->parse_rise(\$subline, $d)){}
1661                elsif($n = $self->parse_fall(\$subline, $d)){}
1662                elsif($n = $self->parse_length(\$subline, $d)){}
1663                elsif($n = $self->parse_wph(\$subline, $d)){}
1664                elsif($n = $self->parse_ipa(\$subline, $d)){}
1665                elsif($n = $self->parse_sm(\$subline, $d)){}
1666                elsif($n = $self->parse_pause(\$subline, $d)){}
1667                elsif($n = $self->parse_hash(\$subline, $d)){}
1668                elsif($n = $self->parse_excl(\$subline, $d)){}
1669                elsif($n = $self->parse_ol(\$subline, $d)){}
1670                elsif($n = $self->parse_sm_laugh(\$subline, $d)){}
1671                else{
1672                    die ('LINE: ' . $self->{LINE_CNT} . 'failed to parse in <un> ...
                        ... xxxx <un> at: '
1673                     . substr($$line, 0, 50));
1674                }
1675                if($n){
1676                    $ui->appendChild($n);
1677                }
1678            }
1679            return $ui;
1680        }
1681        return '';
1682    }
1683
1684    #
1685    # parser: contextual events
1686    #
1687    sub parse_contextual (\$$){
1688        my ($self, $line, $d) = @_;
1689        if(ref($line) eq 'REF'){
1690            $line = $$line;
1691        }
1692        if($$line =~ s/^\{(.+?)\}//){
1693            my $desc = $1;
1694            my $co = $d->createElement('incident');
1695            if($desc =~ s/\(([0-9]+)\)//){
1696                my $dur = $1;
1697                $co->setAttribute('dur',('PT' . $dur . 'S'));
1698            }
1699            $co->setAttribute('voice:desc', $desc);
1700            return $co;
1701        }
1702        return '';
1703    }
1704
1705    #
1706    # parser: punctuation in transcriber notes
1707    #
1708    #Q: what is this? shouldn't it be something like rise or fall??? A:
1709    #should be ok as it is only used in translations (explanations
1710    #unfelicious choice of att name)
1711    sub parse_punct(\$$){
1712        my ($self, $line, $d) = @_;
1713        if(ref($line) eq 'REF'){
1714            $line = $$line;
1715        }
1716        if($$line =~ s/^([.,:;""''\/])//){
1717            return $d->createTextNode($1);
1718        }
1719        return '';
1720    }
1721
```

```
1722 #
1723 # parser: distance / closeness in translations of foreign language
1724 #
1725 sub parse_you_dis_cls(\$$){
1726     my ($self, $line, $d) = @_;
1727     if(ref($line) eq 'REF'){
1728         $line = $$line;
1729     }
1730     if($$line =~ s/^((you\/dis)|(you\/cls))//){
1731         return $d->createTextNode($1);
1732     }
1733     return '';
1734 }
1735
1736 #
1737 # parser: translate language codes from transcript to standard codes
1738 #
1739 sub translate_code($){
1740     my ($self, $short_code) = @_;
1741
1742     if($short_code eq 'xx'){
1743         return 'und';
1744     }
1745
1746     if($self->{USE_DB}){
1747         unless($self->{DB_H}){
1748             $self->connect_db();
1749         }
1750         unless($self->{DB_H}){
1751             die ('Language code Translation: could not connect to DB');
1752         }
1753     }
1754
1755     my $sh = $self->{DB_H}->prepare("SELECT `code` FROM "
1756       . "`language_codes_iso_639_2` WHERE `code_short` = '" . $short_code . "'...
            ...");
1757     $sh->execute();
1758     my $r = $sh->fetchrow_hashref();
1759     if($r){
1760         return $r->{'code'};
1761     }
1762     else{
1763         warn ("Could not translate short_code '" . $short_code . "' to long ...
            ...code");
1764         return $short_code;
1765     }
1766
1767 }
1768
1769 #
1770 # parser: foreign speech
1771 #
1772 sub parse_lx(\$$){
1773     my ($self, $line, $d) = @_;
1774     if(ref($line) eq 'REF'){
1775         $line = $$line;
1776     }
1777     if($$line =~ s/^<(L[1NQ])([a-z]{2,3})>(.+?) <\/\1\2>//){
1778         my $lx = $d->createElement('foreign');
1779         my ($ln1, $code, $subline) = ($1, $2, $3);
1780         $lx->setAttribute('voice:todo', 'TEIEQUIV: type unclean') if $self->{...
            ...DEBUG};
1781         $lx->setAttribute('type', $ln1);
```

```
1782            $lx->setAttribute('xml:lang', $self->translate_code($code));
1783                while(length($subline)>0){
1784                    my $n = 0;
1785                    if($subline =~ s/^\{(.+?)\}//){
1786                        my $subsubline = $1;
1787                        my $trans = '';
1788                            $self->{NO_WORD} = 1;
1789                        while(length($subsubline) > 0){
1790                            if($n = $self->parse_space(\$subsubline, $d)){}
1791                            elsif($n = $self->parse_you_dis_cls(\$subsubline, $d))...
                                ...{}
1792                            elsif($n = $self->parse_punct(\$subsubline, $d)){}
1793                            elsif($n = $self->parse_uncertain(\$subsubline, $d)){}
1794                            elsif($n = $self->parse_wp_nw(\$subsubline, $d)){}
1795                            else{
1796                                die ('LINE: ' . $self->{LINE_CNT}
1797                             . "failed to parse in <${ln1}${code}> {}: "
1798                             . substr($subsubline, 0, 50));
1799                            }
1800                            if($n){
1801                                $trans .= $n->getTextContent();
1802                            }
1803                        }
1804                        if($trans){
1805                            $n = '';
1806                            $lx->setAttribute('voice:translation', $trans);
1807                        }
1808                            $self->{NO_WORD} = 1;
1809                    }
1810                elsif($n = $self->parse_space(\$subline, $d)){}
1811                    elsif($n = $self->parse_wp_stretch(\$subline, $d)){}
1812                    else{
1813                        die ('LINE: ' . $self->{LINE_CNT} . 'failed to parse in L...
                            ...[1N]: \''
1814                            . substr($subline, 0 , 50) . "'\n");
1815                    }
1816                    if($n){
1817                        $lx->appendChild($n);
1818                    }
1819                }
1820            return $lx;
1821        }
1822        return ''
1823    }
1824
1825    #
1826    # parser: speaking-mode laughingly
1827    #
1828    sub parse_sm_laugh(\$$){
1829        my ($self, $line, $d) = @_;
1830        if(ref($line) eq 'REF'){
1831            $line = $$line;
1832        }
1833        if($$line =~ s/^<\@>(.+?) <\/\@>//){
1834            my $subline = $1;
1835            my $id1 = $self->{XMLID_PREFIX} . '_s_' . ++$self->{SHIFT_CNT};
1836            my $id2 = $self->{XMLID_PREFIX} . '_s_' . ++$self->{SHIFT_CNT};
1837            my $ldf = $d->createDocumentFragment();
1838            my $lb = $d->createElement('shift');
1839            $lb->setAttribute('xml:id', $id1);
1840            $lb->setAttribute('corresp', '#'.$id2);
1841            $lb->setAttribute('new', 'laugh');
1842            $lb->setAttribute('feature', 'voice');
```

```
1843            $ldf->appendChild($lb);
1844            while(length($subline)>0){
1845                my $n = 0;
1846                if($n = $self->parse_wp_stretch(\$subline, $d)){}
1847                else{
1848                    die 'LINE ' . $self->{LINE_CNT} . ': failed to parse in <@> ...
                        ... </@>: '
1849                        . substr($subline, 0, 10) . "\n";
1850                    exit(1);
1851                }
1852                if($n){
1853                    $ldf->appendChild($n);
1854                }
1855            }
1856            my $le = $d->createElement('shift');
1857            $le->setAttribute('feature', 'voice');
1858            $le->setAttribute('new', 'neutral');
1859            $le->setAttribute('xml:id',$id2);
1860            $le->setAttribute('corresp','#'.$id1);
1861            $ldf->appendChild($le);
1862            return $ldf;
1863        }
1864        return '';
1865    }
1866
1867    #
1868    # parser: anonymised speech
1869    #
1870    sub parse_anon(\$$){
1871        my ($self, $line, $d) = @_;
1872        if(ref($line) eq 'REF'){
1873            $line = $$line;
1874        }
1875        if($$line =~ s/^(\[.+?\])//){
1876            my $anon = $1;
1877            my $an = $d->createElement('supplied');
1878            $an->setAttribute('reason', 'anonymization');
1879            $an->appendChild($d->createTextNode($anon));
1880            $self->{NO_WORD} = 0;
1881            return $an;
1882        }
1883        return '';
1884    }
1885
1886    #
1887    # parser: reference to transcriber note
1888    #
1889    sub parse_hash(\$$){
1890        my ($self, $line, $d) = @_;
1891        if(ref($line) eq 'REF'){
1892            $line = $$line;
1893        }
1894        if($$line =~ s/^<\#([0-9][0-9]*)>//){
1895            my $h = $d->createElement('ref');
1896            $h->setAttribute('type', 'hash');
1897            $h->setAttribute('target', '#' . $self->{XMLID_PREFIX} . '_h_' . $1);
1898            return $h;
1899        }
1900        return '';
1901    }
1902
1903    #
1904    # parser: transcriber note
```

```
1905 #
1906 sub parse_tn_hash(\$$){
1907     my ($self, $line, $d) = @_;
1908     if(ref($line) eq 'REF'){
1909         $line = $$line;
1910     }
1911     if($$line =~ s/^<\#([0-9]+)>(.*?)<\/\#\1>//s){
1912         my $num = $1;
1913         my $subline = $2;
1914         my $t = $d->createElement('note');
1915         $t->setAttribute('xml:id', $self->{XMLID_PREFIX} . '_h_' . $1 );
1916         while(length($subline) > 0){
1917             my $n = '';
1918             if($self->eat_nl(\$subline)){}
1919             elsif($n = $self->parse_tn_text(\$subline, $d)){}
1920             else{
1921                 die 'LINE ' . $self->{LINE_CNT} . ": failed in <#${num}></#${...
                        ...num}> : "
1922             . substr($subline, 0, 50);
1923             }
1924
1925             if($n){
1926                 $t->appendChild($n);
1927             }
1928         }
1929         return $t;
1930     }
1931 }
1932
1933 #
1934 # parser: reference to transcriber note: level analytic / personalised
1935 #
1936 sub parse_excl_hash_voice(\$$){
1937     my($self, $line, $d) = @_;
1938     if(ref($line) eq 'REF'){
1939         $line = $$line;
1940     }
1941     my $n = '';
1942     if($$line =~ s/^<(?:\!|\#)((?:mlp|sm|tk|ab|ch|hb)(?:_[a-z]{1,4})?)>//){
1943         my $com = $1;
1944         $n = $d->createElement('anchor');
1945         $n->setAttribute('type','marker');
1946         $n->setAttribute('subtype',$com);
1947     }
1948     return $n;
1949 }
1950
1951 #
1952 # parser: reference transcriber note: level analytic.
1953 #
1954 sub parse_excl(\$$){
1955     my ($self, $line, $d) = @_;
1956     if(ref($line) eq 'REF'){
1957         $line = $$line;
1958     }
1959     if($$line =~ s/^<\!(([0-9][0-9]*)?)>//){
1960         my $h = $d->createElement('ref');
1961         $h->setAttribute('type', 'excl');
1962         if($1){
1963             $h->setAttribute('target', '#' . $self->{XMLID_PREFIX} . '_x_' . ...
                    ...$1);
1964         }
1965         return $h;
```

```
1966        }
1967        return '';
1968    }
1969
1970    #
1971    # parser: transcriber note: level analytic
1972    #
1973    sub parse_tn_excl(\$$){
1974        my ($self, $line, $d) = @_;
1975        if(ref($line) eq 'REF'){
1976            $line = $$line;
1977        }
1978        if($$line =~ s/^<!([0-9]+)>(.*?)<\/!\1>//s){
1979            my $num = $1;
1980            my $subline = $2;
1981            my $t = $d->createElement('note');
1982            $t->setAttribute('xml:id', $self->{XMLID_PREFIX} . '_x_' . $1);
1983            while(length($subline) > 0){
1984                my $n = '';
1985                if($self->eat_nl(\$subline)){}
1986                elsif($n = $self->parse_tn_text(\$subline, $d)){}
1987                else{
1988                    die 'LINE ' . $self->{LINE_CNT} . ': failed in <!${num}></!{...
                        ...num}> : '
1989                . substr($subline, 0, 50);
1990                }
1991                if($n){
1992                    $t->appendChild($n);
1993                }
1994            }
1995            return $t;
1996        }
1997        return '';
1998    }
1999
2000    #
2001    # parser: track marker -> indicates position in audio recording
2002    #
2003    sub parse_track(\$$){
2004        my ($self, $line, $d) = @_;
2005        if(ref($line) eq 'REF'){
2006            $line = $$line;
2007        }
2008        if($$line =~ s/^<track +([A-Za-z]{2,3})([0-9]{1,2}(?:\([a-z]\))?)_...
            ...([0-9]{1,2}) >//){
2009            my ($type, $medium, $num) = ($1, $2, $3);
2010            my $t = $d->createElement('voice:track');
2011            $t->setAttribute('type',$type);
2012            $t->setAttribute('medium',$medium);
2013            $t->setAttribute('num',$num);
2014            return $t;
2015        }
2016        return '';
2017    }
2018
2019    #
2020    # parser: spelt word
2021    #
2022    sub parse_spel(\$$){
2023        my ($self, $line, $d) = @_;
2024        if(ref($line) eq 'REF'){
2025                    $line = $$line;
2026        }
```

```
2027    if($$line =~ s/^<spel>(.+?)<\/spel>//){
2028                my $sp_text  = $1;
2029                $sp_text =~ s/^[ \t]*(.+)[ \t]*$/\1/; #strip space
2030
2031                my $sf = $d->createDocumentFragment();
2032
2033                $sf->appendChild($d->createTextNode('␣'));
2034                my $w = $d->createElement('w');
2035                $w->appendChild($d->createTextNode('␣'));
2036
2037                $w->setAttribute('voice:mode', 'spelt');
2038                $sf->appendChild($w);
2039                $sf->appendChild($d->createTextNode('␣'));
2040
2041                my $s = $d->createDocumentFragment();
2042
2043                while(length($sp_text) > 0){
2044                        my $n = '';
2045                        if($sp_text =~ s/^([a-zA-Z])((?:-|\'[sS])?)//){␣#'
2046                                # reset NO_WORD as we enter a word here
2047                                $self->{NO_WORD} = 0;
2048                                my $c = $1;
2049                                my $g = $2 if $2;
2050                                $s->appendChild($d->createTextNode($c));
2051                                if($g){
2052                                        $n = $d->createTextNode($g);
2053                                        warn (($g eq '-'?'truncation':'...
                                                ...genitive') . '␣in␣<spel>:␣"'
2054                                  . $g . '"␣' . $sp_text);
2055                                }
2056                        }
2057                        elsif($sp_text =~ s/^[ \t]+//){
2058                                # TODO replace with milestone
2059                                $n = $d->createTextNode('␣');
2060                        }
2061                        # shift rise fall hash excl to the end
2062                        elsif($n = $self->parse_fall(\$sp_text, $d)){}
2063                        elsif($n = $self->parse_anon(\$sp_text, $d)){
2064                            if($sp_text =~ s/^-//){
2065                                $s->appendChild($n);
2066                                $n = $d->createTextNode('-');
2067                            }
2068                        }
2069                        elsif($n = $self->parse_rise(\$sp_text, $d)){}
2070                        elsif($n = $self->parse_hash(\$sp_text, $d)){}
2071                        elsif($n = $self->parse_excl(\$sp_text, $d)){}
2072                        elsif($n = $self->parse_length(\$sp_text, $d)){}
2073                        elsif($n = $self->parse_track(\$sp_text, $d)){}
2074                        elsif($n = $self->parse_uncertain(\$sp_text, $d)){}
2075                        elsif($n = $self->parse_pause(\$sp_text, $d)){
2076                                warn "pause␣in␣<spel>!␣that␣makes␣sense?";
2077                        }
2078                        elsif($n = $self->parse_ol(\$sp_text, $d)){
2079                                warn "overlap␣in␣<spel>!␣content:␣" . $sp_text...
                                        ...;
2080                        }
2081                        elsif($n = $self->parse_sm_laugh(\$sp_text, $d)){
2082                                warn '!<@>␣in␣<spel>␣content:␣' . $sp_text;
2083                        }
2084                        else{
2085                                die ('LINE:␣' . $self->{LINE_CNT} . "␣failed␣...
                                        ...to␣parse␣in␣<spel>:" .
2086                                                substr($sp_text, 0 , 50));
```

```
2087                            }
2088                            if($n){
2089                                    $s->appendChild($n);
2090                            }
2091                    }
2092                    $w->appendChild($s);
2093                    $w->appendChild($d->createTextNode('␣'));
2094                    $self->{NO_WORD} = 1;
2095                    return $sf;
2096            }
2097            return '';
2098    }
2099
2100    #
2101    # parser: speaker noise
2102    #
2103    sub parse_sn(\$$){
2104            my ($self, $line, $d) = @_;
2105            if(ref($line) eq 'REF'){
2106                    $line = $$line;
2107            }
2108            if($$line =~ s/^<(sighs|squeals|applauds|coughs|clears throat|sneezes|...
                    ...snorts|yawns|whistles)(?: +\(([.0-9]+)\))?>//){
2109                    my $sn = $d->createElement('vocal');
2110                    my $desc = $1;
2111                    my $time = $2?$2:undef;
2112                    if($time && !($time eq '.')){
2113                            $sn->setAttribute('dur', 'PT' . $time . 'S');
2114                            $sn->setAttribute('iterated', 'true');
2115                    }
2116                    $sn->setAttribute('voice:desc', $desc);
2117                    return $sn;
2118            }
2119            elsif($$line =~ s/^<([a-z]+[a-z ]*)( +\(([.0-9]+)\))?>//){
2120                    my $noise = $1;
2121                    my $sn = $d->createElement('vocal');
2122                    $sn->appendChild($d->createComment("Custom␣speaker_noise␣$noise")) if ...
                            ...$self->{DEBUG};
2123                    $sn->setAttribute('type', 'speaker_noise');
2124                    if($3 && !($3 eq '.')){
2125                            $sn->setAttribute('dur', 'PT' . $3 . 'S');
2126                            $sn->setAttribute('iterated', 'true');
2127                    }
2128                    $sn->setAttribute('voice:desc', $noise);
2129                    return $sn;
2130            }
2131            return '';
2132    }
2133
2134    #
2135    # parser: speaking mode -> shift to + shift back to neutral. Explicit linking.
2136    #
2137    sub parse_sm(\$$){
2138            my ($self, $line, $d) = @_;
2139            if(ref($line) eq 'REF'){
2140                    $line = $$line;
2141            }
2142            if($$line =~ s/^<(fast|soft|slow|loud|whispering|sighing|singing|yawning|...
                    ...reading|reading aloud|on phone|imitating)>(.+?) <\/\1>//){
2143                    my $feat = $1;
2144                    my $subline = $2;
2145                    my $new = '';
2146                    my $id1 = $self->{XMLID_PREFIX} . '_s_' . ++$self->{SHIFT_CNT};
```

```
2147        my $id2 = $self->{XMLID_PREFIX} . '_s_' . ++$self->{SHIFT_CNT};
2148
2149        my $sm = $d->createDocumentFragment();
2150        my $so = $d->createElement('shift');
2151        $so->setAttribute('xml:id',$id1);
2152        $so->setAttribute('corresp','#'.$id2);
2153        my $sc = $d->createElement('shift');
2154        $sc->setAttribute('xml:id',$id2);
2155        $sc->setAttribute('corresp','#'.$id1);
2156        if($feat eq 'fast'){
2157            $feat = 'tempo';
2158            $new = 'a';
2159        }
2160        elsif($feat eq 'slow'){
2161            $feat = 'tempo';
2162            $new = 'l';
2163        }
2164        elsif($feat eq 'soft'){
2165            $feat = 'loud';
2166            $new = 'p';
2167        }
2168        elsif($feat eq 'loud'){
2169            $feat = 'loud';
2170            $new = 'f';
2171        }
2172        elsif($feat eq 'whispering'){
2173            $feat = 'voice';
2174            $new = 'whisp';
2175        }
2176        elsif($feat eq 'sighing'){
2177            $feat = 'voice';
2178            $new = 'sigh';
2179        }
2180        elsif($feat eq 'reading'){
2181            $feat = 'voice';
2182            $new = 'reading';
2183        }
2184        elsif($feat eq 'reading␣aloud'){
2185            $feat = 'voice';
2186            $new = 'reading_aloud';
2187        }
2188        elsif($feat eq 'on␣phone'){
2189            $feat = 'voice';
2190            $new = 'phone';
2191        }
2192        elsif($feat eq 'imitating'){
2193            $feat = 'voice';
2194            $new = 'imitating';
2195        }
2196        elsif($feat eq 'singing'){
2197            $feat = 'voice';
2198            $new = 'singing';
2199        }
2200        elsif($feat eq 'yawning'){
2201            $feat = 'voice';
2202            $new = 'yawning';
2203        }
2204        else{
2205            die "Error:␣Hit␣a␣bug␣in␣$self->parse_sm()\n";
2206        }
2207        $so->setAttribute('feature', $feat);
2208        $so->setAttribute('new', $new);
2209        $sm->appendChild($so);
```

```
2210            while(length($subline) > 0){
2211                my $n = '';
2212                if($n = $self->parse_wp_stretch(\$subline, $d)){}
2213                else{
2214                    print STDERR "Failed to parse in <" . $feat . ">: ";
2215                    print STDERR substr($subline, 0, 50);
2216                    die ('LINE: ' . $self->{LINE_CNT} . "died in <" . $feat . ">")...
                        ...;
2217                }
2218                if($n){
2219                    $sm->appendChild($n);
2220                }
2221            }
2222            $sc->setAttribute('feature', $feat);
2223            $sc->setAttribute('new', 'neutral');
2224            $sm->appendChild($sc);
2225            return $sm;
2226        }
2227        if($$line =~ s/^<([a-z][a-z ]*)>(.+?) <\/\1>//){
2228            my $feat = $1;
2229            $feat =~ s/ /_/g;
2230            my $subline = $2;
2231            my $new = '';
2232            my $sm = $d->createDocumentFragment();
2233
2234            my $id1 = $self->{XMLID_PREFIX} . '_s_' . ++$self->{SHIFT_CNT};
2235            my $id2 = $self->{XMLID_PREFIX} . '_s_' . ++$self->{SHIFT_CNT};
2236
2237            my $so = $d->createElement('shift');
2238            $so->setAttribute('xml:id',$id1);
2239            $so->setAttribute('corresp','#'.$id2);
2240            my $sc = $d->createElement('shift');
2241            $sc->setAttribute('xml:id',$id2);
2242            $sc->setAttribute('corresp','#'.$id1);
2243            $sm->appendChild($d->createComment("Custom speaking mode $feat")) if ...
                ...$self->{DEBUG};
2244            $self->status("Custom speaking mode $feat");
2245            $so->setAttribute('feature', 'custom');
2246            $so->setAttribute('new', $feat);
2247            $sm->appendChild($so);
2248            while(length($subline) > 0){
2249                my $n = '';
2250                if($n = $self->parse_wp_stretch(\$subline, $d)){}
2251                else{
2252                    print STDERR "Failed to parse in <" . $feat . ">: ";
2253                    print STDERR substr($subline, 0, 50);
2254                    die ('LINE: ' . $self->{LINE_CNT} . "died in <" . $feat . ">")...
                        ...;
2255                }
2256                if($n){
2257                    $sm->appendChild($n);
2258                }
2259            }
2260            $sc->setAttribute('feature', 'custom');
2261            $sc->setAttribute('new', 'neutral');
2262            $sm->appendChild($sc);
2263            return $sm;
2264        }
2265        return '';
2266    }
2267
2268    #
2269    # parser: overlapping speech
```

```
2270 #
2271 sub parse_ol(\$$){
2272     my ($self, $line, $d) = @_;
2273     if(ref($line) eq 'REF'){
2274         $line = $$line;
2275     }
2276     if($$line =~ s/^<([1-9][0-9]?)>(.*?)<\/\1>//){
2277         my $olnum = $1;
2278         my $subline = $2;
2279 #        print STDERR "OL: " . $subline;
2280         my $ol = $d->createElement('seg');
2281         $ol->setAttribute('type', 'overlap');
2282 
2283         my $id = $self->{XMLID_PREFIX} . '_ol_' . $self->{OVERLAP_CNT}++;
2284         $ol->setAttribute('xml:id', $id);
2285         $ol->setAttribute('n', $olnum);
2286         if(exists($self->{OVERLAP_MAP}->{$olnum})){
2287             $self->{OVERLAP_MAP}->{$olnum} =~ m/_ol_([0-9]+)$/;
2288             my $refol = $1;
2289             $id =~ m/_ol_([0-9]+)$/;
2290             my $ownol = $1;
2291             if($self->{DISAMBIGUATE_OVERLAPS}){
2292                 if($ownol - $refol > $self->{DISAMBIGUATE_OVERLAPS_DEPTH}){
2293                     my $warn_msg = "␣AUTODISAMBIGUATION␣<$olnum>␣$refol:$ownol...
                        ...␣("
2294                 . ($refol - $ownol) . ").␣Trying␣to␣disambiguate,␣please␣check...
                    ...␣";
2295                     $self->status($warn_msg);
2296                     $ol->appendChild($d->createComment('LINE␣' . $self->{...
                        ...LINE_CNT}
2297                 . ':' . $warn_msg )) if $self->{DEBUG}; #.': '. $subline));
2298                     $self->{OVERLAP_MAP}->{$olnum} = $id;
2299                     $self->{OVERLAP_REFNUM}->{$olnum} = 0;
2300                 }
2301                 else{
2302                     $ol->setAttribute('synch', '#' . $self->{OVERLAP_MAP}->{...
                        ...$olnum});
2303                     $self->{OVERLAP_REFNUM}->{$olnum}++;
2304                 }
2305             }
2306             else{
2307                 my $warn_msg = "this␣overlap␣might␣be␣ambiguous␣<$olnum>␣...
                    ...$refol:$ownol."
2308             . "␣NOT␣DISAMBIGUATING";
2309                 $self->status($warn_msg) if $ownol - $refol > $self->{...
                    ...DISAMBIGUATE_OVERLAPS_DEPTH};
2310                 $ol->appendChild($d->createComment($warn_msg .':␣'. $subline))...
                    ... if $self->{DEBUG};
2311                 $ol->setAttribute('synch', '#' . $self->{OVERLAP_MAP}->{$olnum...
                    ...});
2312                 $self->{OVERLAP_REFNUM}->{$olnum}++;
2313             }
2314         }
2315         else{
2316             $self->{OVERLAP_MAP}->{$olnum} = $id;
2317             $self->{OVERLAP_REFNUM}->{$olnum} = 0;
2318         }
2319         while(length($subline) > 0){
2320             my $n = '';
2321             if($n = $self->parse_wp_stretch(\$subline, $d)){}
2322             else{
2323                 die ('LINE:␣' . $self->{LINE_CNT} . "Failed␣parsing␣in␣<$olnum...
                    ...>:␣" .
```

```
2324                     substr($subline, 0, 50));
2325             }
2326
2327             if($n){
2328                 $ol->appendChild($n);
2329             }
2330         }
2331         $self->{OVERLAP_IN_UTTERANCE} = 1;
2332         return $ol;
2333     }
2334     return '';
2335 }
2336
2337 #
2338 # parser: pronunciation variation and coinage
2339 #
2340 sub parse_pvc(\$$){
2341     my ($self, $line, $d) = @_;
2342     if(ref($line) eq 'REF'){
2343         $line = $$line;
2344     }
2345     if($$line =~ s/^<pvc>(.*?)<\/pvc>//){
2346         my $subline = $1;
2347         my $pvc = $d->createElement('voice:pvc');
2348         while (length($subline) > 0){
2349             my $n = '';
2350             if($subline =~ s/^{(.+?)}//){
2351                 my $trans = $1;
2352                 $pvc->setAttribute('comment', $trans);
2353             }
2354             elsif($n = $self->parse_ipa(\$subline, $d)){}
2355             elsif($n = $self->parse_pause(\$subline, $d)){}
2356             elsif($n = $self->parse_uncertain(\$subline, $d)){}
2357             elsif($n = $self->parse_ol(\$subline, $d)){}
2358             elsif($n = $self->parse_wpl(\$subline, $d)){}
2359             elsif($n = $self->parse_wpu(\$subline, $d)){}
2360             elsif($n = $self->parse_wph(\$subline, $d)){}
2361             elsif($n = $self->parse_fall(\$subline, $d)){}
2362             elsif($n = $self->parse_rise(\$subline, $d)){}
2363             elsif($n = $self->parse_space(\$subline, $d)){
2364                     if($n eq 'NO_WORD'){
2365                         next;
2366                     }
2367                 }
2368             elsif($n = $self->parse_length(\$subline, $d)){}
2369             elsif($n = $self->parse_hash(\$subline, $d)){}
2370             elsif($n = $self->parse_excl(\$subline, $d)){}
2371             elsif($n = $self->parse_sm_laugh(\$subline, $d)){}
2372             elsif($n = $self->parse_sm(\$subline, $d)){}
2373             elsif($n = $self->parse_track(\$subline, $d)){
2374                 warn "<track> in <pvc>";
2375             }
2376             else{
2377                 die ('LINE: ' . $self->{LINE_CNT} . "failed to parse in <pvc>"
2378              . substr($subline, 0, 50));
2379             }
2380             if($n){
2381                 $pvc->appendChild($n);
2382             }
2383         }
2384         return $pvc;
2385     }
2386     return '';
```

```
2387 }
2388
2389 #
2390 # parser: international phonetic alphabeth. Used for unintelligible
2391 # speech / pvc
2392 #
2393 sub parse_ipa(\$$){
2394     my ($self, $line, $d) = @_;
2395     if(ref($line) eq 'REF'){
2396         $line = $$line;
2397     }
2398     if($$line =~ s/^<ipa>(.+?)<\/ipa>//){
2399         my $ipacontent = $1;
2400         my $ipa = $d->createElement('seg');
2401         $ipa->setAttribute('type', 'ipa');
2402         $ipa->setAttribute('voice:todo', 'maybe␣restrict␣to␣IPA␣Chars') if ...
                ...$self->{DEBUG};
2403         $ipa->appendChild($d->createTextNode($ipacontent));
2404         return $ipa;
2405     }
2406     return '';
2407 }
2408
2409 #
2410 # parser: ipa word part -> used in ono
2411 #
2412 sub parse_ipa_wp(\$$){
2413     my ($self, $line, $d) = @_;
2414     if(ref($line) eq 'REF'){
2415         $line = $$line;
2416     }
2417
2418     my $ret = '';
2419
2420     if($$line =~ s/^([^>]+)//){
2421         $ret = $d->createTextNode($1);
2422     }
2423
2424     return $ret;
2425 }
2426
2427 #
2428 # parser: onomatopoeic speech
2429 #
2430 sub parse_ono(\$$){
2431     my ($self, $line, $d) = @_;
2432     if(ref($line) eq 'REF'){
2433         $line = $$line;
2434     }
2435     if($$line =~ s/^<ono>(.+?)<\/ono>//){
2436         my $onocontent = $1;
2437         my $ono = $d->createElement('seg');
2438         $ono->setAttribute('type', 'onomatopoeia');
2439         $ono->setAttribute('voice:todo', 'maybe␣restrict␣to␣IPA␣Chars') if ...
                ...$self->{DEBUG};
2440         while(length($onocontent)>0){
2441             my $n = '';
2442             if($n = $self->parse_ipa_wp(\$onocontent, $d)){}
2443             else{
2444                 die ('LINE:␣' . $self->{LINE_CNT} . '␣in␣<ono>:␣' . substr(...
                        ...$$line, 0 , 50));
2445             }
2446             if($n){
```

```
2447                    $ono->appendChild($n);
2448                }
2449            }
2450            return $ono;
2451        }
2452        return '';
2453    }
2454
2455    #
2456    # parser: other continuation --> uses tei:anchor
2457    #
2458    sub parse_othercont(\$$){
2459        my ($self, $line, $d) = @_;
2460        if(ref($line) eq 'REF'){
2461            $line = $$line;
2462        }
2463        if($$line =~ s/^=//){
2464            my $oc = $d->createElement('anchor');
2465            $oc->setAttribute('type', 'other_continuation');
2466            $oc->setAttribute('xml:id', $self->{XMLID_PREFIX} . '_oc_'
2467                              . $self->{OTHER_CONT_CNT});
2468            if($self->{FRESH_UTTERANCE}){
2469                $oc->setAttribute('synch', '#' . $self->{XMLID_PREFIX}
2470                                  . '_oc_' . ($self->{OTHER_CONT_CNT} - 1));
2471                $self->{OTHER_CONT_B_CNT}++;
2472            }
2473            elsif(!$self->{FINISHED_UTTERANCE}){
2474                $oc->setAttribute('synch', '#' . $self->{XMLID_PREFIX} . '_oc_' . ...
                        ...($self->{OTHER_CONT_CNT} + 1));
2475                $self->{OTHER_CONT_C_CNT}++;
2476                $self->{FINISHED_UTTERANCE} = 1;
2477            }
2478            else{
2479                die ('LINE: ' . $self->{LINE_CNT}
2480                 . "neither fresh utterance nor unfinished one: stumbling over = "
2481                 . substr($$line, 0 , 50));
2482            }
2483            $self->{OTHER_CONT_CNT}++;
2484            return $oc;
2485        }
2486        return '';
2487    }
2488
2489    #
2490    # parser: parse speech directed to another speaker
2491    #
2492    sub parse_directed(\$$){
2493        my ($self, $line, $d) = @_;
2494        if(ref($line) eq 'REF'){
2495            $line = $$line;
2496        }
2497        if($$line =~ s/^<to +(S(?:[1-9][0-9]?|(?:X(?:-[mf1-9])?)|S))>(.+?)<\/to ...
                ...+\1>//){
2498            my $who = $1;
2499            my $dir = $d->createElement('voice:to');
2500            $dir->setAttribute('who', ('#' . $self->{XMLID_PREFIX} . '_' . $who));
2501            $dir->setAttribute('voice:todo', 'TEIEQUIV: add element') if $self->{...
                    ...DEBUG};
2502            my $subline = $2;
2503            while(length($subline) > 0){
2504                my $n = '';
2505                if($n = $self->parse_wp_stretch(\$subline, $d)){}
2506                else{
```

```
2507                    die ('LINE: ' . $self->{LINE_CNT} . "Failed parsing <to ${1}>:...
                            ..."
2508                 . substr($subline, 0, 50));
2509                }
2510                if($n){
2511                    $dir->appendChild($n);
2512                }
2513            }
2514            return $dir;
2515        }
2516        return '';
2517    }
2518
2519    ######################################################################
2520    # </routines>
2521    ######################################################################
2522    1;
```

## C.2 VOICE TEI Customisation

The following XML document represents the schemas used for the encoding of VOICE. The document follows the schema for TEI ODD documents (see Burnard2008: chapter 22).

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <TEI xmlns="http://www.tei-c.org/ns/1.0" xml:lang="en"
3       xmlns:rng="http://relaxng.org/ns/structure/1.0">
4    <teiHeader>
5      <fileDesc>
6        <titleStmt>
7          <title>VOICE TEI Customisation</title>
8        </titleStmt>
9        <editionStmt>
10         <edition>built on
11         <date>2009-05-04</date>
12         </edition>
13       </editionStmt>
14       <publicationStmt>
15         <p>unpublished</p>
16       </publicationStmt>
17       <sourceDesc>
18         <p>
19           Generated by merging a manually crafted ODD with one
20           generated with oddbyexample.xsl.
21         </p>
22       </sourceDesc>
23     </fileDesc>
24     <revisionDesc>
25       <change who="SM" when="2010-09-16">att.datable missed @when</change>
26       <change who="SM" when="2010-12-16">moved ipa in unint and pvc to
27       attribute, generated seperate schema for full/anon versions of
28       VOICE</change>
29       <change who="SM">Last modified: <date>$LastChangedDate:
30       2010-12-16 14:10:48 +0100 (Thu, 16 Dec 2010) $</date></change>
31     </revisionDesc>
32   </teiHeader>
33   <text>
34     <body>
35       <divGen type="toc"/>
36       <div>
37         <head>Introduction</head>
```

```
38          <p>
39            The <title>VOICE TEI Customisation</title> is a subset of
40            the TEI Guidelines. Most of this documentation is directly
41            derived from the TEI sources. It is designed to be
42            conformant to the TEI Guidelines, such that it only limits
43            the set of elements and attributes available in the
44            TEI. Therefore, most definitions are deletions of Elements
45            that are not required for VOICE. In some cases it was
46            required to add elements or attributes to elements. These
47            additions are all handled transparently in the namespace
48            <idno
49            type="URL">http://www.univie.ac.at/voice/ns/1.0</idno>.
50          </p>
51        </div>
52        <div>
53          <head>Specification Summary</head>
54          <specGrp xml:id="teiElementsWithExtensions">
55          <moduleRef key="tei"/>
56          <classSpec ident="att.canonical" module="tei" type="atts" mode="change">
57            <attList>
58              <attDef ident="key" mode="delete"/>
59              <attDef ident="ref" mode="delete"/>
60            </attList>
61          </classSpec>
62          <classSpec ident="att.dimensions" module="tei" type="atts" mode="change">
63            <attList>
64              <attDef ident="unit" mode="delete"/>
65              <attDef ident="quantity" mode="delete"/>
66              <attDef ident="extent" mode="delete"/>
67              <attDef ident="atLeast" mode="delete"/>
68              <attDef ident="atMost" mode="delete"/>
69              <attDef ident="min" mode="delete"/>
70              <attDef ident="max" mode="delete"/>
71              <attDef ident="precision" mode="delete"/>
72              <attDef ident="scope" mode="delete"/>
73            </attList>
74          </classSpec>
75          <classSpec ident="att.damaged" module="tei" type="atts" mode="change">
76            <attList>
77              <attDef ident="hand" mode="delete"/>
78              <attDef ident="agent" mode="delete"/>
79              <attDef ident="degree" mode="delete"/>
80              <attDef ident="group" mode="delete"/>
81            </attList>
82          </classSpec>
83          <classSpec ident="att.datable.w3c" module="tei" type="atts" mode="change">
84            <attList>
85              <attDef ident="period" mode="delete"/>
86              <attDef ident="notBefore" mode="delete"/>
87              <attDef ident="notAfter" mode="delete"/>
88            </attList>
89          </classSpec>
90          <classSpec ident="att.declarable" module="tei" type="atts" mode="change">
91            <attList>
92              <attDef ident="default" mode="delete"/>
93            </attList>
94          </classSpec>
95          <classSpec ident="att.declaring" module="tei" type="atts" mode="change">
96            <attList>
97              <attDef ident="decls" mode="delete"/>
98            </attList>
99          </classSpec>
100         <classSpec ident="att.divLike" module="tei" type="atts" mode="change">
```

```
101           <attList>
102             <attDef ident="org" mode="delete"/>
103             <attDef ident="sample" mode="delete"/>
104             <attDef ident="part" mode="delete"/>
105           </attList>
106         </classSpec>
107         <classSpec ident="att.editLike" module="tei" type="atts" mode="change">
108           <attList>
109             <attDef ident="cert" mode="delete"/>
110             <attDef ident="resp" mode="delete"/>
111             <attDef ident="evidence" mode="delete"/>
112             <attDef ident="source" mode="delete"/>
113           </attList>
114         </classSpec>
115         <classSpec ident="att.global" module="tei" type="atts" mode="change">
116           <attList>
117             <attDef ident="rendition" mode="delete"/>
118           </attList>
119         </classSpec>
120         <classSpec ident="att.handFeatures" module="tei" type="atts" mode="change">
121           <attList>
122             <attDef ident="scribe" mode="delete"/>
123             <attDef ident="script" mode="delete"/>
124             <attDef ident="medium" mode="delete"/>
125             <attDef ident="scope" mode="delete"/>
126           </attList>
127         </classSpec>
128         <classSpec ident="att.internetMedia" module="tei" type="atts" mode="change">
129           <attList>
130             <attDef ident="mimeType" mode="delete"/>
131           </attList>
132         </classSpec>
133         <classSpec ident="att.interpLike" module="tei" type="atts" mode="change">
134           <attList>
135             <attDef ident="resp" mode="delete"/>
136             <attDef ident="type" mode="delete"/>
137             <attDef ident="inst" mode="delete"/>
138           </attList>
139         </classSpec>
140         <classSpec ident="att.measurement" module="tei" type="atts" mode="change">
141           <attList>
142             <attDef ident="unit" mode="delete"/>
143             <attDef ident="quantity" mode="delete"/>
144             <attDef ident="commodity" mode="delete"/>
145           </attList>
146         </classSpec>
147         <classSpec ident="att.naming" module="tei" type="atts" mode="change">
148           <attList>
149             <attDef ident="nymRef" mode="delete"/>
150           </attList>
151         </classSpec>
152         <classSpec ident="att.placement" module="tei" type="atts" mode="change">
153           <attList>
154             <attDef ident="place" mode="delete"/>
155           </attList>
156         </classSpec>
157         <classSpec ident="att.segLike" module="tei" type="atts" mode="change">
158           <attList>
159             <attDef ident="part" mode="delete"/>
160           </attList>
161         </classSpec>
162         <classSpec ident="att.sourced" module="tei" type="atts" mode="change">
163           <attList>
```

```
164             <attDef ident="ed" mode="delete"/>
165           </attList>
166         </classSpec>
167         <classSpec ident="att.spanning" module="tei" type="atts" mode="change">
168           <attList>
169             <attDef ident="spanTo" mode="delete"/>
170           </attList>
171         </classSpec>
172         <classSpec ident="att.tableDecoration" module="tei"
173                    type="atts" mode="change">
174           <attList>
175             <attDef ident="role" mode="delete"/>
176             <attDef ident="rows" mode="delete"/>
177             <attDef ident="cols" mode="delete"/>
178           </attList>
179         </classSpec>
180         <classSpec ident="att.timed" module="tei" type="atts" mode="change">
181           <attList>
182             <attDef ident="start" mode="delete"/>
183             <attDef ident="end" mode="delete"/>
184           </attList>
185         </classSpec>
186         <classSpec ident="att.transcriptional" module="tei"
187                    type="atts" mode="change">
188           <attList>
189             <attDef ident="hand" mode="delete"/>
190             <attDef ident="status" mode="delete"/>
191             <attDef ident="seq" mode="delete"/>
192           </attList>
193         </classSpec>
194         <classSpec ident="att.translatable" module="tei" type="atts"
195                    mode="change">
196           <attList>
197             <attDef ident="version" mode="delete"/>
198           </attList>
199         </classSpec>
200         <classSpec ident="att.xmlspace" module="tei" type="atts" mode="change">
201           <attList>
202             <attDef ident="xml:space" mode="delete"/>
203           </attList>
204         </classSpec>
205         <classSpec ident="att.personal" module="tei" type="atts" mode="change">
206           <attList>
207             <attDef ident="full" mode="delete"/>
208             <attDef ident="sort" mode="delete"/>
209           </attList>
210         </classSpec>
211         <moduleRef key="core"/>
212         <moduleRef key="analysis"/>
213         <elementSpec ident="w" mode="change">
214           <content>
215             <zeroOrMore xmlns="http://relaxng.org/ns/structure/1.0">
216               <choice>
217                 <text/>
218                 <ref name="model.gLike"/>
219                 <ref name="seg"/>
220                 <ref name="w"/>
221                 <ref name="m"/>
222                 <ref name="c"/>
223                 <ref name="model.global"/>
224                 <ref name="model.lPart"/>
225                 <ref name="model.highlighted"/>
226                 <ref name="model.pPart.edit"/>
```

```
227             </choice>
228           </zeroOrMore>
229         </content>
230         <attList>
231           <attDef ident="lemma" mode="delete"/>
232           <attDef ident="lemmaRef" mode="delete"/>
233           <attDef ident="n" mode="delete"/>
234           <attDef ident="xml:lang" mode="delete"/>
235           <attDef ident="xml:base" mode="delete"/>
236           <attDef ident="function" mode="delete"/>
237           <attDef ident="type" mode="delete"/>
238           <attDef ident="subtype" mode="delete"/>
239           <attDef usage="opt" ident="mode" mode="add"
240                   ns="http://www.univie.ac.at/voice/ns/1.0">
241             <desc>Describes the mode a word is uttered in</desc>
242             <valList>
243               <valItem ident="spelt"/>
244             </valList>
245           </attDef>
246         </attList>
247       </elementSpec>
248       <elementSpec ident="c" mode="change">
249         <attList>
250           <attDef ident="n" mode="delete"/>
251           <attDef ident="xml:lang" mode="delete"/>
252           <attDef ident="xml:base" mode="delete"/>
253           <attDef ident="subtype" mode="delete"/>
254         </attList>
255       </elementSpec>
256       <elementSpec ident="s" module="analysis" mode="delete"/>
257       <elementSpec ident="cl" module="analysis" mode="delete"/>
258       <elementSpec ident="phr" module="analysis" mode="delete"/>
259       <elementSpec ident="m" module="analysis" mode="delete"/>
260       <elementSpec ident="span" module="analysis" mode="delete"/>
261       <elementSpec ident="spanGrp" module="analysis" mode="delete"/>
262       <elementSpec ident="interp" module="analysis" mode="delete"/>
263       <elementSpec ident="interpGrp" module="analysis" mode="delete"/>
264       <classSpec ident="att.global.analytic" module="analysis"
265                  type="atts" mode="change">
266         <attList>
267           <attDef ident="ana" mode="delete"/>
268         </attList>
269       </classSpec>
270       <elementSpec ident="p" mode="change">
271         <attList>
272           <attDef ident="n" mode="delete"/>
273           <attDef ident="xml:lang" mode="delete"/>
274           <attDef ident="xml:base" mode="delete"/>
275         </attList>
276       </elementSpec>
277       <elementSpec ident="foreign" mode="change">
278         <classes mode="change">
279           <!-- this is not clean -->
280           <memberOf key="att.typed"/>
281         </classes>
282         <attList>
283           <attDef ident="n" mode="delete"/>
284           <attDef ident="xml:base" mode="delete"/>
285           <attDef ident="translation"
286                   ns="http://www.univie.ac.at/voice/ns/1.0">
287             <desc>Translation or comment in English regarding the
288             meaning of the element content</desc>
289             <datatype>
```

```
290             <rng:text/>
291           </datatype>
292         </attDef>
293       </attList>
294     </elementSpec>
295     <elementSpec ident="emph" mode="change">
296       <attList>
297         <attDef ident="n" mode="delete"/>
298         <attDef ident="xml:lang" mode="delete"/>
299         <attDef ident="xml:base" mode="delete"/>
300       </attList>
301     </elementSpec>
302     <elementSpec ident="gap" mode="change">
303       <attList>
304         <attDef ident="hand" mode="delete"/>
305         <attDef ident="agent" mode="delete"/>
306         <attDef ident="n" mode="delete"/>
307         <attDef ident="xml:lang" mode="delete"/>
308         <attDef ident="xml:base" mode="delete"/>
309         <attDef usage="opt" ident="desc" mode="add"
310                 ns="http://www.univie.ac.at/voice/ns/1.0">
311           <datatype>
312             <rng:text/>
313           </datatype>
314         </attDef>
315       </attList>
316     </elementSpec>
317     <elementSpec ident="unclear" mode="change">
318       <attList>
319         <attDef ident="reason" mode="delete"/>
320         <attDef ident="hand" mode="delete"/>
321         <attDef ident="agent" mode="delete"/>
322         <attDef ident="n" mode="delete"/>
323         <attDef ident="xml:lang" mode="delete"/>
324         <attDef ident="xml:base" mode="delete"/>
325       </attList>
326     </elementSpec>
327     <elementSpec ident="name" mode="change">
328       <attList>
329         <attDef ident="n" mode="delete"/>
330         <attDef ident="xml:lang" mode="delete"/>
331         <attDef ident="xml:base" mode="delete"/>
332         <attDef ident="subtype" mode="delete"/>
333       </attList>
334     </elementSpec>
335     <elementSpec ident="address" mode="change">
336       <attList>
337         <attDef ident="n" mode="delete"/>
338         <attDef ident="xml:lang" mode="delete"/>
339         <attDef ident="xml:base" mode="delete"/>
340       </attList>
341     </elementSpec>
342     <elementSpec ident="addrLine" mode="change">
343       <attList>
344         <attDef ident="n" mode="delete"/>
345         <attDef ident="xml:lang" mode="delete"/>
346         <attDef ident="xml:base" mode="delete"/>
347       </attList>
348     </elementSpec>
349     <elementSpec ident="date" mode="change">
350       <attList>
351         <attDef ident="calendar" mode="delete"/>
352         <attDef ident="n" mode="delete"/>
```

```
353             <attDef ident="xml:lang" mode="delete"/>
354             <attDef ident="xml:base" mode="delete"/>
355             <attDef ident="dur" mode="delete"/>
356             <attDef ident="type" mode="delete"/>
357             <attDef ident="subtype" mode="delete"/>
358           </attList>
359         </elementSpec>
360         <elementSpec ident="ref" mode="change">
361           <classes>
362             <memberOf key="model.ptrLike"/>
363             <memberOf key="model.global"/>
364             <memberOf key="att.pointing"/>
365             <memberOf key="att.typed"/>
366             <memberOf key="att.declaring"/>
367           </classes>
368           <attList>
369             <attDef ident="cRef" mode="delete"/>
370             <attDef ident="n" mode="delete"/>
371             <attDef ident="xml:lang" mode="delete"/>
372             <attDef ident="xml:base" mode="delete"/>
373           </attList>
374         </elementSpec>
375         <elementSpec ident="list" mode="change">
376           <attList>
377             <attDef ident="type" mode="change">
378               <valList mode="add" type="closed">
379                 <valItem ident="ordered"/>
380               </valList>
381             </attDef>
382             <attDef ident="n" mode="delete"/>
383             <attDef ident="xml:lang" mode="delete"/>
384             <attDef ident="rend" mode="change">
385               <valList mode="add" type="closed">
386                 <valItem ident="p"/>
387               </valList>
388             </attDef>
389             <attDef ident="xml:base" mode="delete"/>
390           </attList>
391         </elementSpec>
392         <elementSpec ident="item" mode="change">
393           <attList>
394             <attDef ident="n" mode="delete"/>
395             <attDef ident="xml:lang" mode="delete"/>
396             <attDef ident="xml:base" mode="delete"/>
397           </attList>
398         </elementSpec>
399         <elementSpec ident="note" mode="change">
400           <attList>
401             <attDef ident="type" mode="delete"/>
402             <attDef ident="resp" mode="delete"/>
403             <attDef ident="anchored" mode="delete"/>
404             <attDef ident="target" mode="delete"/>
405             <attDef ident="targetEnd" mode="delete"/>
406             <attDef ident="n" mode="delete"/>
407             <attDef ident="xml:lang" mode="delete"/>
408             <attDef ident="xml:base" mode="delete"/>
409           </attList>
410         </elementSpec>
411         <elementSpec ident="analytic" mode="change">
412           <attList>
413             <attDef ident="n" mode="delete"/>
414             <attDef ident="xml:lang" mode="delete"/>
415             <attDef ident="xml:base" mode="delete"/>
```

```
416           </attList>
417         </elementSpec>
418         <elementSpec ident="monogr" mode="change">
419           <attList>
420             <attDef ident="n" mode="delete"/>
421             <attDef ident="xml:lang" mode="delete"/>
422             <attDef ident="xml:base" mode="delete"/>
423           </attList>
424         </elementSpec>
425         <elementSpec ident="author" mode="change">
426           <attList>
427             <attDef ident="n" mode="delete"/>
428             <attDef ident="xml:lang" mode="delete"/>
429             <attDef ident="xml:base" mode="delete"/>
430           </attList>
431         </elementSpec>
432         <elementSpec ident="editor" mode="change">
433           <attList>
434             <attDef ident="role" mode="delete"/>
435             <attDef ident="n" mode="delete"/>
436             <attDef ident="xml:lang" mode="delete"/>
437             <attDef ident="xml:base" mode="delete"/>
438           </attList>
439         </elementSpec>
440         <elementSpec ident="respStmt" mode="change">
441           <attList>
442             <attDef ident="n" mode="delete"/>
443             <attDef ident="xml:lang" mode="delete"/>
444             <attDef ident="xml:base" mode="delete"/>
445           </attList>
446         </elementSpec>
447         <elementSpec ident="resp" mode="change">
448           <attList>
449             <attDef ident="n" mode="delete"/>
450             <attDef ident="xml:lang" mode="delete"/>
451             <attDef ident="xml:base" mode="delete"/>
452           </attList>
453         </elementSpec>
454         <elementSpec ident="title" mode="change">
455           <attList>
456             <attDef ident="level" mode="delete"/>
457             <attDef ident="type" mode="delete"/>
458             <attDef ident="n" mode="delete"/>
459             <attDef ident="xml:lang" mode="delete"/>
460             <attDef ident="xml:base" mode="delete"/>
461           </attList>
462         </elementSpec>
463         <elementSpec ident="imprint" mode="change">
464           <attList>
465             <attDef ident="n" mode="delete"/>
466             <attDef ident="xml:lang" mode="delete"/>
467             <attDef ident="xml:base" mode="delete"/>
468           </attList>
469         </elementSpec>
470         <elementSpec ident="publisher" mode="change">
471           <attList>
472             <attDef ident="n" mode="delete"/>
473             <attDef ident="xml:lang" mode="delete"/>
474             <attDef ident="xml:base" mode="delete"/>
475           </attList>
476         </elementSpec>
477         <elementSpec ident="biblScope" mode="change">
478           <attList>
```

```
479             <attDef ident="type" mode="change">
480               <valList mode="add" type="closed">
481                 <valItem ident="issue"/>
482                 <valItem ident="pp"/>
483                 <valItem ident="vol"/>
484               </valList>
485             </attDef>
486             <attDef ident="n" mode="delete"/>
487             <attDef ident="xml:lang" mode="delete"/>
488             <attDef ident="xml:base" mode="delete"/>
489           </attList>
490         </elementSpec>
491         <elementSpec ident="pubPlace" mode="change">
492           <attList>
493             <attDef ident="n" mode="delete"/>
494             <attDef ident="xml:lang" mode="delete"/>
495             <attDef ident="xml:base" mode="delete"/>
496           </attList>
497         </elementSpec>
498         <elementSpec ident="bibl" mode="change">
499           <attList>
500             <attDef ident="n" mode="delete"/>
501             <attDef ident="xml:lang" mode="delete"/>
502             <attDef ident="xml:base" mode="delete"/>
503             <attDef ident="type" mode="delete"/>
504             <attDef ident="subtype" mode="delete"/>
505           </attList>
506         </elementSpec>
507         <elementSpec ident="biblStruct" mode="change">
508           <attList>
509             <attDef ident="n" mode="delete"/>
510             <attDef ident="xml:lang" mode="delete"/>
511             <attDef ident="xml:base" mode="delete"/>
512             <attDef ident="type" mode="delete"/>
513             <attDef ident="subtype" mode="delete"/>
514           </attList>
515         </elementSpec>
516         <elementSpec ident="listBibl" mode="change">
517           <attList>
518             <attDef ident="n" mode="delete"/>
519             <attDef ident="xml:lang" mode="delete"/>
520             <attDef ident="xml:base" mode="delete"/>
521             <attDef ident="type" mode="delete"/>
522             <attDef ident="subtype" mode="delete"/>
523           </attList>
524         </elementSpec>
525         <elementSpec ident="teiCorpus" mode="change">
526           <attList>
527             <attDef ident="n" mode="delete"/>
528             <attDef ident="xml:lang" mode="delete"/>
529             <attDef ident="xml:base" mode="delete"/>
530           </attList>
531         </elementSpec>
532         <elementSpec ident="hi" module="core" mode="delete"/>
533         <elementSpec ident="distinct" module="core" mode="delete"/>
534         <elementSpec ident="said" module="core" mode="delete"/>
535         <elementSpec ident="quote" module="core" mode="delete"/>
536         <elementSpec ident="q" module="core" mode="delete"/>
537         <elementSpec ident="cit" module="core" mode="delete"/>
538         <elementSpec ident="mentioned" module="core" mode="delete"/>
539         <elementSpec ident="soCalled" module="core" mode="delete"/>
540         <elementSpec ident="desc" module="core" mode="delete"/>
541         <elementSpec ident="gloss" module="core" mode="delete"/>
```

```
542         <elementSpec ident="term" module="core" mode="delete"/>
543         <elementSpec ident="sic" module="core" mode="delete"/>
544         <elementSpec ident="corr" module="core" mode="delete"/>
545         <elementSpec ident="choice" module="core" mode="delete"/>
546         <elementSpec ident="reg" module="core" mode="delete"/>
547         <elementSpec ident="orig" module="core" mode="delete"/>
548         <elementSpec ident="add" module="core" mode="delete"/>
549         <elementSpec ident="del" module="core" mode="delete"/>
550         <elementSpec ident="rs" module="core" mode="delete"/>
551         <elementSpec ident="email" module="core" mode="delete"/>
552         <elementSpec ident="street" module="core" mode="delete"/>
553         <elementSpec ident="postCode" module="core" mode="delete"/>
554         <elementSpec ident="postBox" module="core" mode="delete"/>
555         <elementSpec ident="num" module="core" mode="delete"/>
556         <elementSpec ident="measure" module="core" mode="delete"/>
557         <elementSpec ident="measureGrp" module="core" mode="delete"/>
558         <elementSpec ident="time" module="core" mode="delete"/>
559         <elementSpec ident="abbr" module="core" mode="delete"/>
560         <elementSpec ident="expan" module="core" mode="delete"/>
561         <elementSpec ident="ptr" module="core" mode="delete"/>
562         <elementSpec ident="label" module="core" mode="delete"/>
563         <elementSpec ident="head" module="core" mode="delete"/>
564         <elementSpec ident="headLabel" module="core" mode="delete"/>
565         <elementSpec ident="headItem" module="core" mode="delete"/>
566         <elementSpec ident="index" module="core" mode="delete"/>
567         <elementSpec ident="graphic" module="core" mode="delete"/>
568         <elementSpec ident="binaryObject" module="core" mode="delete"/>
569         <elementSpec ident="milestone" module="core" mode="delete"/>
570         <elementSpec ident="pb" module="core" mode="delete"/>
571         <elementSpec ident="lb" module="core" mode="delete"/>
572         <elementSpec ident="cb" module="core" mode="delete"/>
573         <elementSpec ident="series" module="core" mode="delete"/>
574         <elementSpec ident="meeting" module="core" mode="delete"/>
575         <elementSpec ident="relatedItem" module="core" mode="delete"/>
576         <elementSpec ident="l" module="core" mode="delete"/>
577         <elementSpec ident="lg" module="core" mode="delete"/>
578         <elementSpec ident="sp" module="core" mode="delete"/>
579         <elementSpec ident="speaker" module="core" mode="delete"/>
580         <elementSpec ident="stage" module="core" mode="delete"/>
581         <elementSpec ident="divGen" module="core" mode="delete"/>
582         <moduleRef key="corpus"/>
583         <elementSpec ident="particDesc" mode="change">
584           <attList>
585             <attDef ident="n" mode="delete"/>
586             <attDef ident="xml:lang" mode="delete"/>
587             <attDef ident="xml:base" mode="delete"/>
588           </attList>
589         </elementSpec>
590         <elementSpec ident="settingDesc" mode="change">
591           <attList>
592             <attDef ident="n" mode="delete"/>
593             <attDef ident="xml:lang" mode="delete"/>
594             <attDef ident="xml:base" mode="delete"/>
595           </attList>
596         </elementSpec>
597         <elementSpec ident="setting" mode="change">
598           <attList>
599             <attDef ident="n" mode="delete"/>
600             <attDef ident="xml:lang" mode="delete"/>
601             <attDef ident="xml:base" mode="delete"/>
602             <attDef ident="who" mode="delete"/>
603           </attList>
604         </elementSpec>
```

```
605         <elementSpec ident="locale" mode="change">
606           <attList>
607             <attDef ident="n" mode="delete"/>
608             <attDef ident="xml:lang" mode="delete"/>
609             <attDef ident="xml:base" mode="delete"/>
610           </attList>
611         </elementSpec>
612         <elementSpec ident="activity" mode="change">
613           <attList>
614             <attDef ident="n" mode="delete"/>
615             <attDef ident="xml:lang" mode="delete"/>
616             <attDef ident="xml:base" mode="delete"/>
617           </attList>
618         </elementSpec>
619         <elementSpec ident="textDesc" module="corpus" mode="delete"/>
620         <elementSpec ident="channel" module="corpus" mode="delete"/>
621         <elementSpec ident="constitution" module="corpus" mode="delete"/>
622         <elementSpec ident="derivation" module="corpus" mode="delete"/>
623         <elementSpec ident="domain" module="corpus" mode="delete"/>
624         <elementSpec ident="factuality" module="corpus" mode="delete"/>
625         <elementSpec ident="interaction" module="corpus" mode="delete"/>
626         <elementSpec ident="preparedness" module="corpus" mode="delete"/>
627         <elementSpec ident="purpose" module="corpus" mode="delete"/>
628         <moduleRef key="header"/>
629         <elementSpec ident="teiHeader" mode="change">
630           <attList>
631             <attDef ident="type" mode="change">
632               <valList mode="add" type="closed">
633                 <valItem ident="corpus"/>
634               </valList>
635             </attDef>
636             <attDef ident="n" mode="delete"/>
637             <attDef ident="xml:lang" mode="delete"/>
638             <attDef ident="xml:base" mode="delete"/>
639           </attList>
640         </elementSpec>
641         <elementSpec ident="fileDesc" mode="change">
642           <attList>
643             <attDef ident="n" mode="delete"/>
644             <attDef ident="xml:lang" mode="delete"/>
645             <attDef ident="xml:base" mode="delete"/>
646           </attList>
647         </elementSpec>
648         <elementSpec ident="titleStmt" mode="change">
649           <attList>
650             <attDef ident="n" mode="delete"/>
651             <attDef ident="xml:lang" mode="delete"/>
652             <attDef ident="xml:base" mode="delete"/>
653           </attList>
654         </elementSpec>
655         <elementSpec ident="sponsor" mode="change">
656           <attList>
657             <attDef ident="n" mode="delete"/>
658             <attDef ident="xml:lang" mode="delete"/>
659             <attDef ident="xml:base" mode="delete"/>
660           </attList>
661         </elementSpec>
662         <elementSpec ident="funder" mode="change">
663           <attList>
664             <attDef ident="n" mode="delete"/>
665             <attDef ident="xml:lang" mode="delete"/>
666             <attDef ident="xml:base" mode="delete"/>
667           </attList>
```

```
668         </elementSpec>
669         <elementSpec ident="principal" mode="change">
670           <attList>
671             <attDef ident="n" mode="delete"/>
672             <attDef ident="xml:lang" mode="delete"/>
673             <attDef ident="xml:base" mode="delete"/>
674           </attList>
675         </elementSpec>
676         <elementSpec ident="editionStmt" mode="change">
677           <attList>
678             <attDef ident="n" mode="delete"/>
679             <attDef ident="xml:lang" mode="delete"/>
680             <attDef ident="xml:base" mode="delete"/>
681           </attList>
682         </elementSpec>
683         <elementSpec ident="edition" mode="change">
684           <attList>
685             <attDef ident="n" mode="delete"/>
686             <attDef ident="xml:lang" mode="delete"/>
687             <attDef ident="xml:base" mode="delete"/>
688           </attList>
689         </elementSpec>
690         <elementSpec ident="extent" mode="change">
691           <attList>
692             <attDef ident="n" mode="delete"/>
693             <attDef ident="xml:lang" mode="delete"/>
694             <attDef ident="xml:base" mode="delete"/>
695           </attList>
696         </elementSpec>
697         <elementSpec ident="publicationStmt" mode="change">
698           <attList>
699             <attDef ident="n" mode="delete"/>
700             <attDef ident="xml:lang" mode="delete"/>
701             <attDef ident="xml:base" mode="delete"/>
702           </attList>
703         </elementSpec>
704         <elementSpec ident="distributor" mode="change">
705           <attList>
706             <attDef ident="n" mode="delete"/>
707             <attDef ident="xml:lang" mode="delete"/>
708             <attDef ident="xml:base" mode="delete"/>
709           </attList>
710         </elementSpec>
711         <elementSpec ident="idno" mode="change">
712           <attList>
713             <attDef ident="type" mode="delete"/>
714             <attDef ident="n" mode="delete"/>
715             <attDef ident="xml:lang" mode="delete"/>
716             <attDef ident="xml:base" mode="delete"/>
717           </attList>
718         </elementSpec>
719         <elementSpec ident="availability" mode="change">
720           <attList>
721             <attDef ident="status" mode="change">
722               <valList mode="add" type="closed">
723                 <valItem ident="free"/>
724               </valList>
725             </attDef>
726             <attDef ident="n" mode="delete"/>
727             <attDef ident="xml:lang" mode="delete"/>
728           </attList>
729         </elementSpec>
730         <elementSpec ident="notesStmt" mode="change">
```

```
731           <attList>
732             <attDef ident="n" mode="delete"/>
733             <attDef ident="xml:lang" mode="delete"/>
734             <attDef ident="xml:base" mode="delete"/>
735           </attList>
736         </elementSpec>
737         <elementSpec ident="sourceDesc" mode="change">
738           <attList>
739             <attDef ident="n" mode="delete"/>
740             <attDef ident="xml:lang" mode="delete"/>
741             <attDef ident="xml:base" mode="delete"/>
742           </attList>
743         </elementSpec>
744         <elementSpec ident="encodingDesc" mode="change">
745           <attList>
746             <attDef ident="n" mode="delete"/>
747             <attDef ident="xml:lang" mode="delete"/>
748             <attDef ident="xml:base" mode="delete"/>
749           </attList>
750         </elementSpec>
751         <elementSpec ident="projectDesc" mode="change">
752           <attList>
753             <attDef ident="n" mode="delete"/>
754             <attDef ident="xml:lang" mode="delete"/>
755             <attDef ident="xml:base" mode="delete"/>
756           </attList>
757         </elementSpec>
758         <elementSpec ident="samplingDecl" mode="change">
759           <attList>
760             <attDef ident="n" mode="delete"/>
761             <attDef ident="xml:lang" mode="delete"/>
762             <attDef ident="xml:base" mode="delete"/>
763           </attList>
764         </elementSpec>
765         <elementSpec ident="editorialDecl" mode="change">
766           <attList>
767             <attDef ident="n" mode="delete"/>
768             <attDef ident="xml:lang" mode="delete"/>
769             <attDef ident="xml:base" mode="delete"/>
770           </attList>
771         </elementSpec>
772         <elementSpec ident="correction" mode="change">
773           <attList>
774             <attDef ident="status" mode="change">
775               <valList mode="add" type="closed">
776                 <valItem ident="high"/>
777               </valList>
778             </attDef>
779             <attDef ident="method" mode="delete"/>
780             <attDef ident="n" mode="delete"/>
781             <attDef ident="xml:lang" mode="delete"/>
782             <attDef ident="xml:base" mode="delete"/>
783           </attList>
784         </elementSpec>
785         <elementSpec ident="normalization" mode="change">
786           <attList>
787             <attDef ident="source" mode="delete"/>
788             <attDef ident="method" mode="delete"/>
789             <attDef ident="n" mode="delete"/>
790             <attDef ident="xml:lang" mode="delete"/>
791             <attDef ident="xml:base" mode="delete"/>
792           </attList>
793         </elementSpec>
```

```
794         <elementSpec ident="segmentation" mode="change">
795           <attList>
796             <attDef ident="n" mode="delete"/>
797             <attDef ident="xml:lang" mode="delete"/>
798             <attDef ident="xml:base" mode="delete"/>
799           </attList>
800         </elementSpec>
801         <elementSpec ident="tagsDecl" mode="change">
802           <attList>
803             <attDef ident="n" mode="delete"/>
804             <attDef ident="xml:lang" mode="delete"/>
805             <attDef ident="xml:base" mode="delete"/>
806           </attList>
807         </elementSpec>
808         <elementSpec ident="tagUsage" mode="change">
809           <attList>
810             <attDef ident="withId" mode="delete"/>
811             <attDef ident="render" mode="delete"/>
812             <attDef ident="n" mode="delete"/>
813             <attDef ident="xml:lang" mode="delete"/>
814             <attDef ident="xml:base" mode="delete"/>
815           </attList>
816         </elementSpec>
817         <elementSpec ident="namespace" mode="change">
818           <attList>
819             <attDef ident="n" mode="delete"/>
820             <attDef ident="xml:lang" mode="delete"/>
821             <attDef ident="xml:base" mode="delete"/>
822           </attList>
823         </elementSpec>
824         <elementSpec ident="classDecl" mode="change">
825           <attList>
826             <attDef ident="n" mode="delete"/>
827             <attDef ident="xml:lang" mode="delete"/>
828             <attDef ident="xml:base" mode="delete"/>
829           </attList>
830         </elementSpec>
831         <elementSpec ident="taxonomy" mode="change">
832           <attList>
833             <attDef ident="n" mode="delete"/>
834             <attDef ident="xml:lang" mode="delete"/>
835             <attDef ident="xml:base" mode="delete"/>
836           </attList>
837         </elementSpec>
838         <elementSpec ident="category" mode="change">
839           <attList>
840             <attDef ident="xml:lang" mode="delete"/>
841             <attDef ident="xml:base" mode="delete"/>
842           </attList>
843         </elementSpec>
844         <elementSpec ident="catDesc" mode="change">
845           <attList>
846             <attDef ident="n" mode="delete"/>
847             <attDef ident="xml:lang" mode="delete"/>
848             <attDef ident="xml:base" mode="delete"/>
849           </attList>
850         </elementSpec>
851         <elementSpec ident="profileDesc" mode="change">
852           <attList>
853             <attDef ident="n" mode="delete"/>
854             <attDef ident="xml:lang" mode="delete"/>
855             <attDef ident="xml:base" mode="delete"/>
856           </attList>
```

```
857         </elementSpec>
858         <elementSpec ident="creation" mode="change">
859           <attList>
860             <attDef ident="n" mode="delete"/>
861             <attDef ident="xml:lang" mode="delete"/>
862             <attDef ident="xml:base" mode="delete"/>
863           </attList>
864         </elementSpec>
865         <elementSpec ident="langUsage" mode="change">
866           <attList>
867             <attDef ident="n" mode="delete"/>
868             <attDef ident="xml:lang" mode="delete"/>
869             <attDef ident="xml:base" mode="delete"/>
870           </attList>
871         </elementSpec>
872         <elementSpec ident="language" mode="change">
873           <attList>
874             <attDef ident="usage" mode="delete"/>
875             <attDef ident="n" mode="delete"/>
876             <attDef ident="xml:lang" mode="delete"/>
877             <attDef ident="xml:base" mode="delete"/>
878           </attList>
879         </elementSpec>
880         <elementSpec ident="textClass" mode="change">
881           <attList>
882             <attDef ident="n" mode="delete"/>
883             <attDef ident="xml:lang" mode="delete"/>
884             <attDef ident="xml:base" mode="delete"/>
885           </attList>
886         </elementSpec>
887         <elementSpec ident="catRef" mode="change">
888           <attList>
889             <attDef ident="scheme" mode="delete"/>
890             <attDef ident="n" mode="delete"/>
891             <attDef ident="xml:lang" mode="delete"/>
892             <attDef ident="xml:base" mode="delete"/>
893           </attList>
894         </elementSpec>
895         <elementSpec ident="revisionDesc" mode="change">
896           <attList>
897             <attDef ident="n" mode="delete"/>
898             <attDef ident="xml:lang" mode="delete"/>
899             <attDef ident="xml:base" mode="delete"/>
900           </attList>
901         </elementSpec>
902         <elementSpec ident="change" mode="change">
903           <attList>
904             <attDef ident="xml:lang" mode="delete"/>
905             <attDef ident="xml:base" mode="delete"/>
906           </attList>
907         </elementSpec>
908         <elementSpec ident="authority" module="header" mode="delete"/>
909         <elementSpec ident="seriesStmt" module="header" mode="delete"/>
910         <elementSpec ident="biblFull" module="header" mode="delete"/>
911         <elementSpec ident="quotation" module="header" mode="delete"/>
912         <elementSpec ident="hyphenation" module="header" mode="delete"/>
913         <elementSpec ident="stdVals" module="header" mode="delete"/>
914         <elementSpec ident="interpretation" module="header" mode="delete"/>
915         <elementSpec ident="rendition" module="header" mode="delete"/>
916         <elementSpec ident="refsDecl" module="header" mode="delete"/>
917         <elementSpec ident="cRefPattern" module="header" mode="delete"/>
918         <elementSpec ident="refState" module="header" mode="delete"/>
919         <elementSpec ident="appInfo" module="header" mode="delete"/>
```

```
920         <elementSpec ident="application" module="header" mode="delete"/>
921         <elementSpec ident="handNote" module="header" mode="delete"/>
922         <elementSpec ident="keywords" module="header" mode="delete"/>
923         <elementSpec ident="classCode" module="header" mode="delete"/>
924         <elementSpec ident="typeNote" module="header" mode="delete"/>
925         <elementSpec ident="geoDecl" module="header" mode="delete"/>
926         <moduleRef key="linking"/>
927         <elementSpec ident="seg" mode="change">
928           <attList>
929             <attDef ident="xml:lang" mode="delete"/>
930             <attDef ident="xml:base" mode="delete"/>
931             <attDef ident="function" mode="delete"/>
932             <attDef ident="subtype" mode="delete"/>
933           </attList>
934         </elementSpec>
935         <elementSpec ident="link" module="linking" mode="delete"/>
936         <elementSpec ident="linkGrp" module="linking" mode="delete"/>
937         <elementSpec ident="ab" module="linking" mode="delete"/>
938         <elementSpec ident="when" module="linking" mode="delete"/>
939         <elementSpec ident="timeline" module="linking" mode="delete"/>
940         <elementSpec ident="join" module="linking" mode="delete"/>
941         <elementSpec ident="joinGrp" module="linking" mode="delete"/>
942         <elementSpec ident="alt" module="linking" mode="delete"/>
943         <elementSpec ident="altGrp" module="linking" mode="delete"/>
944         <classSpec ident="att.global.linking" module="linking"
945                    type="atts" mode="change">
946           <attList>
947             <attDef ident="copyOf" mode="delete"/>
948             <attDef ident="next" mode="delete"/>
949             <attDef ident="prev" mode="delete"/>
950             <attDef ident="exclude" mode="delete"/>
951             <attDef ident="select" mode="delete"/>
952           </attList>
953         </classSpec>
954         <classSpec ident="att.pointing" module="linking" type="atts"
955                    mode="change">
956           <attList>
957             <attDef ident="type" mode="delete"/>
958             <attDef ident="evaluate" mode="delete"/>
959           </attList>
960         </classSpec>
961         <classSpec ident="att.pointing.group" module="linking"
962                    type="atts" mode="change">
963           <attList>
964             <attDef ident="domains" mode="delete"/>
965             <attDef ident="targFunc" mode="delete"/>
966           </attList>
967         </classSpec>
968         <moduleRef key="namesdates"/>
969         <elementSpec ident="orgName" mode="change">
970           <attList>
971             <attDef ident="n" mode="delete"/>
972             <attDef ident="xml:lang" mode="delete"/>
973             <attDef ident="xml:base" mode="delete"/>
974             <attDef ident="from" mode="delete"/>
975             <attDef ident="to" mode="delete"/>
976             <attDef ident="type" mode="delete"/>
977             <attDef ident="subtype" mode="delete"/>
978           </attList>
979         </elementSpec>
980         <elementSpec ident="persName" mode="change">
981           <attList>
982             <attDef ident="n" mode="delete"/>
```

```
983             <attDef ident="xml:lang" mode="delete"/>
984             <attDef ident="xml:base" mode="delete"/>
985             <attDef ident="from" mode="delete"/>
986             <attDef ident="to" mode="delete"/>
987             <attDef ident="type" mode="delete"/>
988             <attDef ident="subtype" mode="delete"/>
989           </attList>
990         </elementSpec>
991         <elementSpec ident="age" mode="change">
992           <attList>
993             <attDef ident="n" mode="delete"/>
994             <attDef ident="xml:lang" mode="delete"/>
995             <attDef ident="xml:base" mode="delete"/>
996             <attDef ident="from" mode="delete"/>
997             <attDef ident="to" mode="delete"/>
998           </attList>
999         </elementSpec>
1000        <elementSpec ident="langKnowledge" mode="change">
1001          <attList>
1002            <attDef ident="tags" mode="delete"/>
1003            <attDef ident="n" mode="delete"/>
1004            <attDef ident="xml:lang" mode="delete"/>
1005            <attDef ident="xml:base" mode="delete"/>
1006            <attDef ident="from" mode="delete"/>
1007            <attDef ident="to" mode="delete"/>
1008          </attList>
1009        </elementSpec>
1010        <elementSpec ident="langKnown" mode="change">
1011          <attList>
1012            <attDef ident="n" mode="delete"/>
1013            <attDef ident="xml:lang" mode="delete"/>
1014            <attDef ident="xml:base" mode="delete"/>
1015            <attDef ident="from" mode="delete"/>
1016            <attDef ident="to" mode="delete"/>
1017          </attList>
1018        </elementSpec>
1019        <elementSpec ident="listPerson" mode="change">
1020          <attList>
1021            <attDef ident="n" mode="delete"/>
1022            <attDef ident="xml:lang" mode="delete"/>
1023            <attDef ident="xml:base" mode="delete"/>
1024            <attDef ident="subtype" mode="delete"/>
1025          </attList>
1026        </elementSpec>
1027        <elementSpec ident="occupation" mode="change">
1028          <attList>
1029            <attDef ident="scheme" mode="delete"/>
1030            <attDef ident="code" mode="delete"/>
1031            <attDef ident="n" mode="delete"/>
1032            <attDef ident="xml:lang" mode="delete"/>
1033            <attDef ident="xml:base" mode="delete"/>
1034            <attDef ident="from" mode="delete"/>
1035            <attDef ident="to" mode="delete"/>
1036          </attList>
1037        </elementSpec>
1038        <elementSpec ident="relationGrp" mode="change">
1039          <attList>
1040            <attDef ident="n" mode="delete"/>
1041            <attDef ident="xml:lang" mode="delete"/>
1042            <attDef ident="xml:base" mode="delete"/>
1043            <attDef ident="type" mode="delete"/>
1044            <attDef ident="subtype" mode="delete"/>
1045          </attList>
```

```
1046        </elementSpec>
1047        <elementSpec ident="person" mode="change">
1048          <attList>
1049            <attDef ident="role" mode="change">
1050              <valList mode="add" type="closed">
1051                <valItem ident="audience"/>
1052                <valItem ident="chair"/>
1053                <valItem ident="coordinator"/>
1054                <valItem ident="interviewee"/>
1055                <valItem ident="interviewer"/>
1056                <valItem ident="non-participant"/>
1057                <valItem ident="participant"/>
1058                <valItem ident="presenter"/>
1059                <valItem ident="researcher"/>
1060                <valItem ident="student"/>
1061                <valItem ident="teacher"/>
1062                <valItem ident="translator"/>
1063              </valList>
1064            </attDef>
1065            <attDef ident="age" mode="delete"/>
1066            <attDef ident="n" mode="delete"/>
1067            <attDef ident="xml:lang" mode="delete"/>
1068            <attDef ident="xml:base" mode="delete"/>
1069          </attList>
1070        </elementSpec>
1071        <elementSpec ident="personGrp" mode="change">
1072          <attList>
1073            <attDef ident="role" mode="change">
1074              <valList mode="add" type="closed">
1075                <valItem ident="audience"/>
1076                <valItem ident="interactants"/>
1077                <valItem ident="speakers"/>
1078              </valList>
1079            </attDef>
1080            <attDef ident="sex" mode="delete"/>
1081            <attDef ident="age" mode="delete"/>
1082            <attDef ident="n" mode="delete"/>
1083            <attDef ident="xml:lang" mode="delete"/>
1084            <attDef ident="xml:base" mode="delete"/>
1085          </attList>
1086        </elementSpec>
1087        <elementSpec ident="relation" mode="change">
1088          <attList>
1089            <attDef ident="type" mode="change">
1090              <valList mode="add" type="closed">
1091                <valItem ident="acquaintedness"/>
1092                <valItem ident="power"/>
1093              </valList>
1094            </attDef>
1095            <attDef ident="name" mode="change">
1096              <valList mode="add" type="closed">
1097                <valItem ident="acquainted"/>
1098                <valItem ident="acquaintedness_unknown"/>
1099                <valItem ident="power_relations_unknown"/>
1100                <valItem ident="predominantly_acquainted"/>
1101                <valItem ident="predominantly_unacquainted"/>
1102                <valItem ident="fairly_asymmetrical"/>
1103                <valItem ident="fairly_symmetrical"/>
1104                <valItem ident="unacquainted"/>
1105              </valList>
1106            </attDef>
1107            <attDef ident="active" mode="delete"/>
1108            <attDef ident="passive" mode="delete"/>
```

```
1109            <attDef ident="n" mode="delete"/>
1110            <attDef ident="xml:lang" mode="delete"/>
1111            <attDef ident="xml:base" mode="delete"/>
1112            <attDef ident="from" mode="delete"/>
1113            <attDef ident="to" mode="delete"/>
1114          </attList>
1115        </elementSpec>
1116        <elementSpec ident="sex" mode="change">
1117          <attList>
1118            <attDef ident="n" mode="delete"/>
1119            <attDef ident="xml:lang" mode="delete"/>
1120            <attDef ident="xml:base" mode="delete"/>
1121            <attDef ident="from" mode="delete"/>
1122            <attDef ident="to" mode="delete"/>
1123          </attList>
1124        </elementSpec>
1125        <elementSpec ident="surname" module="namesdates" mode="delete"/>
1126        <elementSpec ident="forename" module="namesdates" mode="delete"/>
1127        <elementSpec ident="genName" module="namesdates" mode="delete"/>
1128        <elementSpec ident="nameLink" module="namesdates" mode="delete"/>
1129        <elementSpec ident="addName" module="namesdates" mode="delete"/>
1130        <elementSpec ident="roleName" module="namesdates" mode="delete"/>
1131        <elementSpec ident="placeName" module="namesdates" mode="delete"/>
1132        <elementSpec ident="bloc" module="namesdates" mode="delete"/>
1133        <elementSpec ident="country" module="namesdates" mode="delete"/>
1134        <elementSpec ident="region" module="namesdates" mode="delete"/>
1135        <elementSpec ident="district" module="namesdates" mode="delete"/>
1136        <elementSpec ident="settlement" module="namesdates" mode="delete"/>
1137        <elementSpec ident="offset" module="namesdates" mode="delete"/>
1138        <elementSpec ident="geogName" module="namesdates" mode="delete"/>
1139        <elementSpec ident="geogFeat" module="namesdates" mode="delete"/>
1140        <elementSpec ident="affiliation" module="namesdates" mode="delete"/>
1141        <elementSpec ident="birth" module="namesdates" mode="delete"/>
1142        <elementSpec ident="climate" module="namesdates" mode="delete"/>
1143        <elementSpec ident="death" module="namesdates" mode="delete"/>
1144        <elementSpec ident="education" module="namesdates" mode="delete"/>
1145        <elementSpec ident="event" module="namesdates" mode="delete"/>
1146        <elementSpec ident="faith" module="namesdates" mode="delete"/>
1147        <elementSpec ident="floruit" module="namesdates" mode="delete"/>
1148        <elementSpec ident="geo" module="namesdates" mode="delete"/>
1149        <elementSpec ident="listOrg" module="namesdates" mode="delete"/>
1150        <elementSpec ident="listEvent" module="namesdates" mode="delete"/>
1151        <elementSpec ident="listPlace" module="namesdates" mode="delete"/>
1152        <elementSpec ident="location" module="namesdates" mode="delete"/>
1153        <elementSpec ident="nationality" module="namesdates" mode="delete"/>
1154        <elementSpec ident="org" module="namesdates" mode="delete"/>
1155        <elementSpec ident="place" module="namesdates" mode="delete"/>
1156        <elementSpec ident="population" module="namesdates" mode="delete"/>
1157        <elementSpec ident="residence" module="namesdates" mode="delete"/>
1158        <elementSpec ident="socecStatus" module="namesdates" mode="delete"/>
1159        <elementSpec ident="state" module="namesdates" mode="delete"/>
1160        <elementSpec ident="terrain" module="namesdates" mode="delete"/>
1161        <elementSpec ident="trait" module="namesdates" mode="delete"/>
1162        <elementSpec ident="nym" module="namesdates" mode="delete"/>
1163        <elementSpec ident="listNym" module="namesdates" mode="delete"/>
1164        <classSpec ident="att.datable.iso" module="namesdates"
1165                   type="atts" mode="change">
1166          <attList>
1167            <attDef ident="when-iso" mode="delete"/>
1168            <attDef ident="notBefore-iso" mode="delete"/>
1169            <attDef ident="notAfter-iso" mode="delete"/>
1170            <attDef ident="from-iso" mode="delete"/>
1171            <attDef ident="to-iso" mode="delete"/>
```

```
1172          </attList>
1173        </classSpec>
1174        <classSpec ident="att.duration.iso" module="namesdates"
1175                   type="atts" mode="change">
1176          <attList>
1177            <attDef ident="dur-iso" mode="delete"/>
1178          </attList>
1179        </classSpec>
1180        <moduleRef key="spoken"/>
1181        <elementSpec ident="scriptStmt" mode="change">
1182          <attList>
1183            <attDef ident="n" mode="delete"/>
1184            <attDef ident="xml:lang" mode="delete"/>
1185            <attDef ident="xml:base" mode="delete"/>
1186          </attList>
1187        </elementSpec>
1188        <elementSpec ident="recordingStmt" mode="change">
1189          <attList>
1190            <attDef ident="n" mode="delete"/>
1191            <attDef ident="xml:lang" mode="delete"/>
1192            <attDef ident="xml:base" mode="delete"/>
1193          </attList>
1194        </elementSpec>
1195        <elementSpec ident="recording" mode="change">
1196          <attList>
1197            <attDef ident="type" mode="change">
1198              <valList mode="add" type="closed">
1199                <valItem ident="audio"/>
1200              </valList>
1201            </attDef>
1202            <attDef ident="n" mode="delete"/>
1203            <attDef ident="xml:lang" mode="delete"/>
1204            <attDef ident="xml:base" mode="delete"/>
1205          </attList>
1206        </elementSpec>
1207        <elementSpec ident="equipment" mode="change">
1208          <attList>
1209            <attDef ident="n" mode="delete"/>
1210            <attDef ident="xml:lang" mode="delete"/>
1211            <attDef ident="xml:base" mode="delete"/>
1212          </attList>
1213        </elementSpec>
1214        <elementSpec ident="u" mode="change">
1215          <attList>
1216            <attDef ident="trans" mode="delete"/>
1217            <attDef ident="n" mode="delete"/>
1218            <attDef ident="xml:lang" mode="delete"/>
1219            <attDef ident="xml:base" mode="delete"/>
1220            <attDef ident="dur" mode="delete"/>
1221          </attList>
1222        </elementSpec>
1223        <elementSpec ident="pause" mode="change">
1224          <attList>
1225            <attDef ident="n" mode="delete"/>
1226            <attDef ident="xml:lang" mode="delete"/>
1227            <attDef ident="xml:base" mode="delete"/>
1228            <attDef ident="type" mode="delete"/>
1229            <attDef ident="subtype" mode="delete"/>
1230            <attDef ident="who" mode="delete"/>
1231          </attList>
1232        </elementSpec>
1233        <elementSpec ident="vocal" mode="change">
1234          <classes mode="replace">
```

```
1235            <memberOf key="model.global.spoken"/>
1236            <memberOf key="att.timed"/>
1237            <memberOf key="att.ascribed"/>
1238            <!-- not clean -->
1239            <memberOf key="att.typed"/>
1240          </classes>
1241          <attList>
1242            <attDef ident="n" mode="delete"/>
1243            <attDef ident="xml:lang" mode="delete"/>
1244            <attDef ident="xml:base" mode="delete"/>
1245            <attDef ident="who" mode="delete"/>
1246            <attDef usage="opt" ident="syl" mode="add"
1247                    ns="http://www.univie.ac.at/voice/ns/1.0">
1248              <desc>Specifies the approx. count of syllables</desc>
1249              <datatype maxOccurs="1">
1250                <rng:ref name="data.count"/>
1251              </datatype>
1252            </attDef>
1253            <attDef usage="opt" ident="desc" mode="add"
1254                    ns="http://www.univie.ac.at/voice/ns/1.0">
1255              <datatype>
1256                <rng:text/>
1257              </datatype>
1258            </attDef>
1259
1260          </attList>
1261        </elementSpec>
1262        <elementSpec ident="incident" mode="change">
1263          <attList>
1264            <attDef ident="n" mode="delete"/>
1265            <attDef ident="xml:lang" mode="delete"/>
1266            <attDef ident="xml:base" mode="delete"/>
1267            <attDef ident="type" mode="delete"/>
1268            <attDef ident="subtype" mode="delete"/>
1269            <attDef ident="who" mode="delete"/>
1270            <attDef usage="opt" ident="desc" mode="add"
1271                    ns="http://www.univie.ac.at/voice/ns/1.0">
1272              <datatype>
1273                <rng:text/>
1274              </datatype>
1275            </attDef>
1276          </attList>
1277        </elementSpec>
1278        <elementSpec ident="shift" mode="change">
1279          <attList>
1280            <attDef ident="feature" mode="change">
1281              <valList mode="add" type="closed">
1282                <valItem ident="custom"/>
1283                <valItem ident="loud"/>
1284                <valItem ident="tempo"/>
1285                <valItem ident="voice"/>
1286              </valList>
1287            </attDef>
1288            <attDef ident="new" mode="change">
1289              <valList mode="add" type="closed">
1290                <valItem ident="a"/>
1291                <valItem ident="f"/>
1292                <valItem ident="high"/>
1293                <valItem ident="imitating"/>
1294                <valItem ident="l"/>
1295                <valItem ident="laugh"/>
1296                <valItem ident="neutral"/>
1297                <valItem ident="not"/>
```

```
1298                <valItem ident="p"/>
1299                <valItem ident="phone"/>
1300                <valItem ident="reading"/>
1301                <valItem ident="reading_aloud"/>
1302                <valItem ident="sigh"/>
1303                <valItem ident="singing"/>
1304                <valItem ident="speaking"/>
1305                <valItem ident="voice"/>
1306                <valItem ident="whisp"/>
1307                <valItem ident="with"/>
1308                <valItem ident="yawning"/>
1309                <valItem ident="speaking_with_a_high_voice"/>
1310              </valList>
1311            </attDef>
1312            <attDef ident="n" mode="delete"/>
1313            <attDef ident="xml:lang" mode="delete"/>
1314            <attDef ident="xml:base" mode="delete"/>
1315            <attDef ident="who" mode="delete"/>
1316          </attList>
1317        </elementSpec>
1318        <elementSpec ident="broadcast" module="spoken" mode="delete"/>
1319        <elementSpec ident="kinesic" module="spoken" mode="delete"/>
1320        <elementSpec ident="writing" module="spoken" mode="delete"/>
1321        <moduleRef key="tagdocs"/>
1322        <elementSpec ident="att" mode="change">
1323          <attList>
1324            <attDef ident="scheme" mode="delete"/>
1325            <attDef ident="n" mode="delete"/>
1326            <attDef ident="xml:lang" mode="delete"/>
1327            <attDef ident="xml:base" mode="delete"/>
1328          </attList>
1329        </elementSpec>
1330        <elementSpec ident="gi" mode="change">
1331          <attList>
1332            <attDef ident="scheme" mode="delete"/>
1333            <attDef ident="n" mode="delete"/>
1334            <attDef ident="xml:lang" mode="delete"/>
1335            <attDef ident="xml:base" mode="delete"/>
1336          </attList>
1337        </elementSpec>
1338        <elementSpec ident="val" mode="change">
1339          <attList>
1340            <attDef ident="n" mode="delete"/>
1341            <attDef ident="xml:lang" mode="delete"/>
1342            <attDef ident="xml:base" mode="delete"/>
1343          </attList>
1344        </elementSpec>
1345        <elementSpec ident="code" module="tagdocs" mode="delete"/>
1346        <elementSpec ident="eg" module="tagdocs" mode="delete"/>
1347        <elementSpec ident="egXML" module="tagdocs" mode="delete"/>
1348        <elementSpec ident="ident" module="tagdocs" mode="delete"/>
1349        <elementSpec ident="tag" module="tagdocs" mode="delete"/>
1350        <elementSpec ident="specList" module="tagdocs" mode="delete"/>
1351        <elementSpec ident="specDesc" module="tagdocs" mode="delete"/>
1352        <elementSpec ident="moduleRef" module="tagdocs" mode="delete"/>
1353        <elementSpec ident="moduleSpec" module="tagdocs" mode="delete"/>
1354        <elementSpec ident="schemaSpec" module="tagdocs" mode="delete"/>
1355        <elementSpec ident="specGrp" module="tagdocs" mode="delete"/>
1356        <elementSpec ident="specGrpRef" module="tagdocs" mode="delete"/>
1357        <elementSpec ident="stringVal" module="tagdocs" mode="delete"/>
1358        <elementSpec ident="elementSpec" module="tagdocs" mode="delete"/>
1359        <elementSpec ident="classSpec" module="tagdocs" mode="delete"/>
1360        <elementSpec ident="macroSpec" module="tagdocs" mode="delete"/>
```

```
1361        <elementSpec ident="remarks" module="tagdocs" mode="delete"/>
1362        <elementSpec ident="listRef" module="tagdocs" mode="delete"/>
1363        <elementSpec ident="exemplum" module="tagdocs" mode="delete"/>
1364        <elementSpec ident="classes" module="tagdocs" mode="delete"/>
1365        <elementSpec ident="memberOf" module="tagdocs" mode="delete"/>
1366        <elementSpec ident="equiv" module="tagdocs" mode="delete"/>
1367        <elementSpec ident="altIdent" module="tagdocs" mode="delete"/>
1368        <elementSpec ident="content" module="tagdocs" mode="delete"/>
1369        <elementSpec ident="attList" module="tagdocs" mode="delete"/>
1370        <elementSpec ident="attDef" module="tagdocs" mode="delete"/>
1371        <elementSpec ident="attRef" module="tagdocs" mode="delete"/>
1372        <elementSpec ident="datatype" module="tagdocs" mode="delete"/>
1373        <elementSpec ident="defaultVal" module="tagdocs" mode="delete"/>
1374        <elementSpec ident="valDesc" module="tagdocs" mode="delete"/>
1375        <elementSpec ident="valItem" module="tagdocs" mode="delete"/>
1376        <elementSpec ident="valList" module="tagdocs" mode="delete"/>
1377        <classSpec ident="att.identified" module="tagdocs" type="atts"
1378                   mode="change">
1379          <attList>
1380            <attDef ident="ident" mode="delete"/>
1381            <attDef ident="predeclare" mode="delete"/>
1382            <attDef ident="module" mode="delete"/>
1383            <attDef ident="mode" mode="delete"/>
1384          </attList>
1385        </classSpec>
1386        <moduleRef key="textstructure"/>
1387        <elementSpec ident="TEI" mode="change">
1388          <attList>
1389            <attDef ident="n" mode="delete"/>
1390          </attList>
1391        </elementSpec>
1392        <elementSpec ident="text" mode="change">
1393          <attList>
1394            <attDef ident="n" mode="delete"/>
1395            <attDef ident="xml:lang" mode="delete"/>
1396            <attDef ident="xml:base" mode="delete"/>
1397            <attDef ident="type" mode="delete"/>
1398            <attDef ident="subtype" mode="delete"/>
1399          </attList>
1400        </elementSpec>
1401        <elementSpec ident="body" mode="change">
1402          <attList>
1403            <attDef ident="n" mode="delete"/>
1404            <attDef ident="xml:lang" mode="delete"/>
1405            <attDef ident="xml:base" mode="delete"/>
1406          </attList>
1407        </elementSpec>
1408        <elementSpec ident="div" mode="change">
1409          <attList>
1410            <attDef ident="n" mode="delete"/>
1411            <attDef ident="xml:lang" mode="delete"/>
1412            <attDef ident="xml:base" mode="delete"/>
1413            <attDef ident="subtype" mode="delete"/>
1414            <attDef usage="opt" ident="end"
1415                    ns="http://www.univie.ac.at/voice/ns/1.0">
1416              <datatype>
1417                <rng:ref name="data.temporal.iso"/>
1418              </datatype>
1419            </attDef>
1420            <attDef usage="opt" ident="start"
1421                    ns="http://www.univie.ac.at/voice/ns/1.0">
1422              <datatype>
1423                <rng:ref name="data.temporal.iso"/>
```

```
1424              </datatype>
1425            </attDef>
1426            <attDef usage="opt" ident="medium_type"
1427                    ns="http://www.univie.ac.at/voice/ns/1.0">
1428              <datatype>
1429                <rng:ref name="data.enumerated"/>
1430              </datatype>
1431            </attDef>
1432            <attDef usage="opt" ident="end_medium_number"
1433                    ns="http://www.univie.ac.at/voice/ns/1.0">
1434              <datatype>
1435                <rng:ref name="data.key"/>
1436              </datatype>
1437            </attDef>
1438            <attDef usage="opt" ident="start_medium_number"
1439                    ns="http://www.univie.ac.at/voice/ns/1.0">
1440              <datatype>
1441                <rng:ref name="data.key"/>
1442              </datatype>
1443            </attDef>
1444            <attDef usage="opt" ident="end_track"
1445                    ns="http://www.univie.ac.at/voice/ns/1.0">
1446              <datatype>
1447                <rng:ref name="data.count"/>
1448              </datatype>
1449            </attDef>
1450            <attDef usage="opt" ident="start_track"
1451                    ns="http://www.univie.ac.at/voice/ns/1.0">
1452              <datatype>
1453                <rng:ref name="data.count"/>
1454              </datatype>
1455            </attDef>
1456
1457          </attList>
1458        </elementSpec>
1459        <elementSpec ident="group" module="textstructure" mode="delete"/>
1460        <elementSpec ident="floatingText" module="textstructure" mode="delete"/>
1461        <elementSpec ident="div1" module="textstructure" mode="delete"/>
1462        <elementSpec ident="div2" module="textstructure" mode="delete"/>
1463        <elementSpec ident="div3" module="textstructure" mode="delete"/>
1464        <elementSpec ident="div4" module="textstructure" mode="delete"/>
1465        <elementSpec ident="div5" module="textstructure" mode="delete"/>
1466        <elementSpec ident="div6" module="textstructure" mode="delete"/>
1467        <elementSpec ident="div7" module="textstructure" mode="delete"/>
1468        <elementSpec ident="trailer" module="textstructure" mode="delete"/>
1469        <elementSpec ident="byline" module="textstructure" mode="delete"/>
1470        <elementSpec ident="dateline" module="textstructure" mode="delete"/>
1471        <elementSpec ident="argument" module="textstructure" mode="delete"/>
1472        <elementSpec ident="epigraph" module="textstructure" mode="delete"/>
1473        <elementSpec ident="opener" module="textstructure" mode="delete"/>
1474        <elementSpec ident="closer" module="textstructure" mode="delete"/>
1475        <elementSpec ident="salute" module="textstructure" mode="delete"/>
1476        <elementSpec ident="signed" module="textstructure" mode="delete"/>
1477        <elementSpec ident="postscript" module="textstructure" mode="delete"/>
1478        <elementSpec ident="titlePage" module="textstructure" mode="delete"/>
1479        <elementSpec ident="docTitle" module="textstructure" mode="delete"/>
1480        <elementSpec ident="titlePart" module="textstructure" mode="delete"/>
1481        <elementSpec ident="docAuthor" module="textstructure" mode="delete"/>
1482        <elementSpec ident="imprimatur" module="textstructure" mode="delete"/>
1483        <elementSpec ident="docEdition" module="textstructure" mode="delete"/>
1484        <elementSpec ident="docImprint" module="textstructure" mode="delete"/>
1485        <elementSpec ident="docDate" module="textstructure" mode="delete"/>
1486        <elementSpec ident="front" module="textstructure" mode="delete"/>
```

```
1487        <elementSpec ident="back" module="textstructure" mode="delete"/>
1488        <moduleRef key="transcr"/>
1489        <elementSpec ident="supplied" mode="change">
1490          <attList>
1491            <attDef ident="n" mode="delete"/>
1492            <attDef ident="xml:lang" mode="delete"/>
1493            <attDef ident="xml:base" mode="delete"/>
1494            <attDef usage="opt" ident="ipa" mode="add"
1495                    ns="http://www.univie.ac.at/voice/ns/1.0">
1496              <desc> Transcription of the item in the International
1497              Phonetic Alphabet (IPA) </desc>
1498              <datatype minOccurs="1" maxOccurs="unbounded">
1499                <rng:data type="token"/>
1500              </datatype>
1501            </attDef>
1502          </attList>
1503        </elementSpec>
1504        <elementSpec ident="facsimile" module="transcr" mode="delete"/>
1505        <elementSpec ident="surface" module="transcr" mode="delete"/>
1506        <elementSpec ident="zone" module="transcr" mode="delete"/>
1507        <elementSpec ident="addSpan" module="transcr" mode="delete"/>
1508        <elementSpec ident="damage" module="transcr" mode="delete"/>
1509        <elementSpec ident="damageSpan" module="transcr" mode="delete"/>
1510        <elementSpec ident="delSpan" module="transcr" mode="delete"/>
1511        <elementSpec ident="ex" module="transcr" mode="delete"/>
1512        <elementSpec ident="fw" module="transcr" mode="delete"/>
1513        <elementSpec ident="handNotes" module="transcr" mode="delete"/>
1514        <elementSpec ident="handShift" module="transcr" mode="delete"/>
1515        <elementSpec ident="am" module="transcr" mode="delete"/>
1516        <elementSpec ident="restore" module="transcr" mode="delete"/>
1517        <elementSpec ident="space" module="transcr" mode="delete"/>
1518        <elementSpec ident="subst" module="transcr" mode="delete"/>
1519        <classSpec ident="att.global.facs" module="transcr"
1520                   type="atts"
1521                   mode="change">
1522          <attList>
1523            <attDef ident="facs" mode="delete"/>
1524          </attList>
1525        </classSpec>
1526        <classSpec ident="att.coordinated" module="transcr"
1527                   type="atts"mode="change">
1528          <attList>
1529            <attDef ident="ulx" mode="delete"/>
1530            <attDef ident="uly" mode="delete"/>
1531            <attDef ident="lrx" mode="delete"/>
1532            <attDef ident="lry" mode="delete"/>
1533          </attList>
1534        </classSpec>
1535        </specGrp>
1536
1537        <specGrp xml:id="anonSpecificElements">
1538        <elementSpec ident="anchor" mode="change">
1539          <attList>
1540            <attDef ident="n" mode="delete"/>
1541            <attDef ident="xml:lang" mode="delete"/>
1542            <attDef ident="xml:base" mode="delete"/>
1543            <attDef ident="subtype" mode="delete"/>
1544          </attList>
1545        </elementSpec>
1546        </specGrp>
1547
1548        <specGrp xml:id="fullTextSpecificElements">
1549        <elementSpec ident="anchor" mode="change">
```

```
1550          <attList>
1551            <attDef ident="n" mode="delete"/>
1552            <attDef ident="xml:lang" mode="delete"/>
1553            <attDef ident="xml:base" mode="delete"/>
1554          </attList>
1555        </elementSpec>
1556        </specGrp>
1557
1558        <specGrp xml:id="voiceElements">
1559        <elementSpec ident="track" mode="add"
1560                     ns="http://www.univie.ac.at/voice/ns/1.0">
1561          <desc>indicating the beginning of a new track.</desc>
1562          <classes>
1563            <memberOf key="model.global"/>
1564            <memberOf key="att.global"/>
1565            <memberOf key="att.typed"/>
1566          </classes>
1567          <content>
1568            <rng:empty/>
1569          </content>
1570          <attList>
1571            <attDef usage="opt" ident="medium">
1572              <datatype><rng:ref name="data.key"/></datatype>
1573            </attDef>
1574            <attDef usage="opt" ident="num">
1575              <datatype><rng:ref name="data.key"/></datatype>
1576            </attDef>
1577          </attList>
1578        </elementSpec>
1579
1580        <elementSpec ident="pvc" mode="add"
1581                     ns="http://www.univie.ac.at/voice/ns/1.0">
1582          <desc>Indicating a Pronunciation Variation and Coinage as
1583          described in the VOICE Transcription Conventions</desc>
1584          <classes>
1585            <memberOf key="model.phrase"/>
1586            <memberOf key="att.global"/>
1587          </classes>
1588          <content>
1589            <rng:ref xmlns:rng="http://relaxng.org/ns/structure/1.0"
1590                     name="macro.paraContent"/>
1591          </content>
1592          <attList>
1593            <attDef usage="opt" ident="comment">
1594              <datatype><rng:text/></datatype>
1595            </attDef>
1596            <attDef usage="opt" ident="ipa" mode="add"
1597                    ns="http://www.univie.ac.at/voice/ns/1.0">
1598              <desc> Transcription of the item in the International
1599              Phonetic Alphabet (IPA) </desc>
1600              <datatype minOccurs="1" maxOccurs="unbounded">
1601                <rng:data type="token"/>
1602              </datatype>
1603            </attDef>
1604          </attList>
1605        </elementSpec>
1606
1607        <elementSpec ident="to" mode="add"
1608                     ns="http://www.univie.ac.at/voice/ns/1.0">
1609          <desc>Indicating a stretch of speach directed to a
1610          person</desc>
1611          <classes>
1612            <memberOf key="model.segLike"/>
```

```
1613              <memberOf key="att.segLike"/>
1614              <memberOf key="att.ascribed"/>
1615            </classes>
1616            <content>
1617              <rng:ref name="macro.paraContent"/>
1618            </content>
1619          </elementSpec>
1620          </specGrp>
1621        </div>
1622        <div>
1623          <head>Schema Details</head>
1624        <schemaSpec ident="VOICESchema" start="TEI teiCorpus">
1625          <specGrpRef target="#teiElementsWithExtensions"/>
1626          <specGrpRef target="#anonSpecificElements"/>
1627          <specGrpRef target="#voiceElements"/>
1628        </schemaSpec>
1629        <schemaSpec ident="VOICEFullTexts" start="TEI">
1630          <specGrpRef target="#teiElementsWithExtensions"/>
1631          <specGrpRef target="#fullTextSpecificElements"/>
1632          <specGrpRef target="#voiceElements"/>
1633        </schemaSpec>
1634      </div>
1635      </body>
1636    </text>
1637 </TEI>
```

## C.3 VOICE Schema

The following schema is derived from the TEI customisation in the previous section of the appendix using the TEI tool Roma. It is automatically generated and includes therefore parts of the documentation from the TEI Guidelines.

```
1  default namespace = "http://www.tei-c.org/ns/Examples"
2  namespace a = "http://relaxng.org/ns/compatibility/annotations/1.0"
3  namespace voice = "http://www.univie.ac.at/voice/ns/1.0"
4  namespace tei = "http://www.tei-c.org/ns/1.0"
5  namespace rng = "http://relaxng.org/ns/structure/1.0"
6  namespace sch = "http://purl.oclc.org/dsdl/schematron"
7  namespace xlink = "http://www.w3.org/1999/xlink"
8
9  # Schema generated from ODD source 2010-09-16T16:28:32Z.
10 # Edition: 1.7.0. Last updated on July 6th 2010.
11 # Edition Location: http://www.tei-c.org/Vault/P5/1.7.0/
12
13 macro.paraContent =
14   (text | model.gLike | model.phrase | model.inter | model.global)*
15 macro.limitedContent = (text | model.limitedPhrase | model.inter)*
16 macro.phraseSeq = (text | model.gLike | model.phrase | model.global)*
17 macro.phraseSeq.limited = (text | model.limitedPhrase | model.global)*
18 macro.specialPara =
19   (text
20    | model.gLike
21    | model.phrase
22    | model.inter
23    | model.divPart
24    | model.global)*
25 macro.xtext = (text | model.gLike)*
26 macro.anyXML =
27   element * - (tei:* | egXML) {
28     attribute * { text }*,
29     (text | macro.anyXML)*
```

```
30    }
31  att.ascribed.attributes = att.ascribed.attribute.who
32  att.ascribed.attribute.who =
33
34    ## indicates the person, or group of people, to whom the element
35    ## content is ascribed.
36    attribute who {
37      list { xsd:anyURI, xsd:anyURI* }
38    }?
39  att.canonical.attributes = empty
40  att.ranging.attributes =
41    att.ranging.attribute.atLeast,
42    att.ranging.attribute.atMost,
43    att.ranging.attribute.min,
44    att.ranging.attribute.max
45  att.ranging.attribute.atLeast =
46
47    ## gives a minimum estimated value for the approximate measurement.
48    attribute atLeast {
49      xsd:double
50      | xsd:token { pattern = "(\-?[\d]+/\-?[\d]+)" }
51      | xsd:decimal
52    }?
53  att.ranging.attribute.atMost =
54
55    ## gives a maximum estimated value for the approximate measurement.
56    attribute atMost {
57      xsd:double
58      | xsd:token { pattern = "(\-?[\d]+/\-?[\d]+)" }
59      | xsd:decimal
60    }?
61  att.ranging.attribute.min =
62
63    ## where the measurement summarizes more than one observation or a
64    ## range, supplies the minimum value observed.
65    attribute min {
66      xsd:double
67      | xsd:token { pattern = "(\-?[\d]+/\-?[\d]+)" }
68      | xsd:decimal
69    }?
70  att.ranging.attribute.max =
71
72    ## where the measurement summarizes more than one observation or a
73    ## range, supplies the maximum value observed.
74    attribute max {
75      xsd:double
76      | xsd:token { pattern = "(\-?[\d]+/\-?[\d]+)" }
77      | xsd:decimal
78    }?
79  att.dimensions.attributes = att.ranging.attributes
80  att.datable.w3c.attributes =
81    att.datable.w3c.attribute.when,
82    att.datable.w3c.attribute.from,
83    att.datable.w3c.attribute.to
84  att.datable.w3c.attribute.when =
85
86    ## supplies the value of the date or time in a standard form,
87    ## e.g. yyyy-mm-dd.
88    attribute when {
89      xsd:date
90      | xsd:gYear
91      | xsd:gMonth
92      | xsd:gDay
```

```
93      | xsd:gYearMonth
94      | xsd:gMonthDay
95      | xsd:time
96      | xsd:dateTime
97    }?
98  att.datable.w3c.attribute.from =
99
100   ## indicates the starting point of the period in standard form,
101   ## e.g. yyyy-mm-dd.
102   attribute from {
103     xsd:date
104     | xsd:gYear
105     | xsd:gMonth
106     | xsd:gDay
107     | xsd:gYearMonth
108     | xsd:gMonthDay
109     | xsd:time
110     | xsd:dateTime
111   }?
112 att.datable.w3c.attribute.to =
113
114   ## indicates the ending point of the period in standard
115   ## form, e.g. yyyy-mm-dd.
116   attribute to {
117     xsd:date
118     | xsd:gYear
119     | xsd:gMonth
120     | xsd:gDay
121     | xsd:gYearMonth
122     | xsd:gMonthDay
123     | xsd:time
124     | xsd:dateTime
125   }?
126 att.declarable.attributes = empty
127 att.declaring.attributes = empty
128 att.divLike.attributes = empty
129 att.docStatus.attributes = att.docStatus.attribute.status
130 att.docStatus.attribute.status =
131
132   ## describes the status of a document either currently or, when
133   ## associated with a dated element, at the time indicated.  Sample
134   ## values include: 1] candidate; 2] recommendation; 3] submitted; 4]
135   ## approved; 5] deprecated; 6] proposed; 7] cleared; 8] embargoed;
136   ## 9] draft; 10] frozen; 11] expired; 12] unfinished; 13] draft; 14]
137   ## galley; 15] published; 16] withdrawn; 17] expired
138   [ a:defaultValue = "draft" ] attribute status { xsd:Name }?
139 att.duration.w3c.attributes = att.duration.w3c.attribute.dur
140 att.duration.w3c.attribute.dur =
141
142   ## (duration) indicates the length of this element in time.
143   attribute dur { xsd:duration }?
144 att.responsibility.attributes =
145   att.responsibility.attribute.cert, att.responsibility.attribute.resp
146 att.responsibility.attribute.cert =
147
148   ## (certainty) signifies the degree of certainty associated with the
149   ## intervention or interpretation.
150   attribute cert { "high" | "medium" | "low" | "unknown" }?
151 att.responsibility.attribute.resp =
152
153   ## (responsible party) indicates the agency responsible for the
154   ## intervention or interpretation, for example an editor or
155   ## transcriber.
```

```
156     attribute resp {
157       list { xsd:anyURI, xsd:anyURI* }
158     }?
159   att.editLike.attributes =
160     att.dimensions.attributes, att.responsibility.attributes
161   att.global.attributes =
162     att.global.linking.attributes,
163     att.global.analytic.attributes,
164     att.global.facs.attributes,
165     att.global.attribute.xmlid,
166     att.global.attribute.n,
167     att.global.attribute.xmllang,
168     att.global.attribute.rend,
169     att.global.attribute.xmlbase,
170     att.global.attribute.xmlspace
171   att.global.attribute.xmlid =
172
173     ## (identifier) provides a unique identifier for the element bearing
174     ## the attribute.
175     attribute xml:id { xsd:ID }?
176   att.global.attribute.n =
177
178     ## (number) gives a number (or other label) for an element, which is
179     ## not necessarily unique within the document.
180     attribute n {
181       list {
182         xsd:token { pattern = "(\p{L}|\p{N}|\p{P}|\p{S})+" },
183         xsd:token { pattern = "(\p{L}|\p{N}|\p{P}|\p{S})+" }*
184       }
185     }?
186   att.global.attribute.xmllang =
187
188     ## (language) indicates the language of the element content using a
189     ## tag generated according to BCP 47
190     attribute xml:lang { xsd:language }?
191   att.global.attribute.rend =
192
193     ## (rendition) indicates how the element in question was rendered or
194     ## presented in the source text.
195     attribute rend {
196       list {
197         xsd:token { pattern = "(\p{L}|\p{N}|\p{P}|\p{S})+" },
198         xsd:token { pattern = "(\p{L}|\p{N}|\p{P}|\p{S})+" }*
199       }
200     }?
201   att.global.attribute.xmlbase =
202
203     ## provides a base URI reference with which applications can resolve
204     ## relative URI references into absolute URI references.
205     attribute xml:base { xsd:anyURI }?
206   att.global.attribute.xmlspace =
207
208     ## signals an intention about how white space should be managed by
209     ## applications.
210     attribute xml:space {
211
212       ## the processor should treat white space according to the default
213       ## XML white space handling rules
214       "default"
215       |
216         ## the processor should preserve unchanged any and all white
217         ## space in the source
218         "preserve"
```

```
219     }?
220   att.naming.attributes =
221     att.canonical.attributes, att.naming.attribute.role
222   att.naming.attribute.role =
223
224     ## may be used to specify further information about the entity
225     ## referenced by this name, for example the occupation of a person,
226     ## or the status of a place.
227     attribute role { xsd:Name }?
228   att.placement.attributes = empty
229   att.typed.attributes =
230     att.typed.attribute.type, att.typed.attribute.subtype
231   att.typed.attribute.type =
232
233     ## characterizes the element in some sense, using any convenient
234     ## classification scheme or typology.
235     attribute type { xsd:Name }?
236   att.typed.attribute.subtype =
237
238     ## provides a sub-categorization of the element, if needed
239     attribute subtype { xsd:Name }?
240   att.pointing.attributes = att.pointing.attribute.target
241   att.pointing.attribute.target =
242
243     ## specifies the destination of the reference by supplying one or
244     ## more URI References
245     attribute target {
246       list { xsd:anyURI, xsd:anyURI* }
247     }?
248   att.readFrom.attributes = att.readFrom.attribute.source
249   att.readFrom.attribute.source =
250
251     ## specifies the source from which declarations and definitions for
252     ## the components of the object being defined may be obtained.
253     attribute source { xsd:anyURI }?
254   att.segLike.attributes = att.segLike.attribute.function
255   att.segLike.attribute.function =
256
257     ## characterizes the function of the segment.
258     attribute function { xsd:Name }?
259   att.timed.attributes = att.duration.w3c.attributes
260   model.nameLike.agent = name | orgName | persName
261   model.nameLike.agent_alternation = name | orgName | persName
262   model.nameLike.agent_sequence = name, orgName, persName
263   model.nameLike.agent_sequenceOptional = name?, orgName?, persName?
264   model.nameLike.agent_sequenceOptionalRepeatable =
265     name*, orgName*, persName*
266   model.nameLike.agent_sequenceRepeatable = name+, orgName+, persName+
267   model.segLike = w | c | pc | seg | to
268   model.segLike_alternation = w | c | pc | seg | to
269   model.segLike_sequence = w, c, pc, seg, to
270   model.segLike_sequenceOptional = w?, c?, pc?, seg?, to?
271   model.segLike_sequenceOptionalRepeatable = w*, c*, pc*, seg*, to*
272   model.segLike_sequenceRepeatable = w+, c+, pc+, seg+, to+
273   model.hiLike = notAllowed
274   model.hiLike_alternation = notAllowed
275   model.hiLike_sequence = empty
276   model.hiLike_sequenceOptional = empty
277   model.hiLike_sequenceOptionalRepeatable = empty
278   model.hiLike_sequenceRepeatable = notAllowed
279   model.emphLike = foreign | emph | title
280   model.emphLike_alternation = foreign | emph | title
281   model.emphLike_sequence = foreign, emph, title
```

```
282 model.emphLike_sequenceOptional = foreign?, emph?, title?
283 model.emphLike_sequenceOptionalRepeatable = foreign*, emph*, title*
284 model.emphLike_sequenceRepeatable = foreign+, emph+, title+
285 model.highlighted = model.hiLike | model.emphLike
286 model.highlighted_alternation =
287   model.hiLike_alternation | model.emphLike_alternation
288 model.highlighted_sequence =
289   model.hiLike_sequence, model.emphLike_sequence
290 model.highlighted_sequenceOptional =
291   model.hiLike_sequenceOptional?, model.emphLike_sequenceOptional?
292 model.highlighted_sequenceOptionalRepeatable =
293   model.hiLike_sequenceOptionalRepeatable*,
294   model.emphLike_sequenceOptionalRepeatable*
295 model.highlighted_sequenceRepeatable =
296   model.hiLike_sequenceRepeatable+, model.emphLike_sequenceRepeatable+
297 model.dateLike = date
298 model.dateLike_alternation = date
299 model.dateLike_sequence = date
300 model.dateLike_sequenceOptional = date?
301 model.dateLike_sequenceOptionalRepeatable = date*
302 model.dateLike_sequenceRepeatable = date+
303 model.pPart.transcriptional = unclear | supplied | surplus
304 model.pPart.transcriptional_alternation = unclear | supplied | surplus
305 model.pPart.transcriptional_sequence = unclear, supplied, surplus
306 model.pPart.transcriptional_sequenceOptional =
307   unclear?, supplied?, surplus?
308 model.pPart.transcriptional_sequenceOptionalRepeatable =
309   unclear*, supplied*, surplus*
310 model.pPart.transcriptional_sequenceRepeatable =
311   unclear+, supplied+, surplus+
312 model.pPart.edit = model.pPart.transcriptional
313 model.pPart.edit_alternation = model.pPart.transcriptional_alternation
314 model.pPart.edit_sequence = model.pPart.transcriptional_sequence
315 model.pPart.edit_sequenceOptional =
316   model.pPart.transcriptional_sequenceOptional?
317 model.pPart.edit_sequenceOptionalRepeatable =
318   model.pPart.transcriptional_sequenceOptionalRepeatable*
319 model.pPart.edit_sequenceRepeatable =
320   model.pPart.transcriptional_sequenceRepeatable+
321 model.ptrLike = ref
322 model.ptrLike_alternation = ref
323 model.ptrLike_sequence = ref
324 model.ptrLike_sequenceOptional = ref?
325 model.ptrLike_sequenceOptionalRepeatable = ref*
326 model.ptrLike_sequenceRepeatable = ref+
327 model.lPart = notAllowed
328 model.lPart_alternation = notAllowed
329 model.lPart_sequence = empty
330 model.lPart_sequenceOptional = empty
331 model.lPart_sequenceOptionalRepeatable = empty
332 model.lPart_sequenceRepeatable = notAllowed
333 model.milestoneLike = anchor
334 model.milestoneLike_alternation = anchor
335 model.milestoneLike_sequence = anchor
336 model.milestoneLike_sequenceOptional = anchor?
337 model.milestoneLike_sequenceOptionalRepeatable = anchor*
338 model.milestoneLike_sequenceRepeatable = anchor+
339 model.gLike = notAllowed
340 model.oddRef = classRef | elementRef | macroRef
341 model.oddRef_alternation = classRef | elementRef | macroRef
342 model.oddRef_sequence = classRef, elementRef, macroRef
343 model.oddRef_sequenceOptional = classRef?, elementRef?, macroRef?
344 model.oddRef_sequenceOptionalRepeatable =
```

```
345   classRef*, elementRef*, macroRef*
346 model.oddRef_sequenceRepeatable = classRef+, elementRef+, macroRef+
347 model.phrase.xml = att | gi | val
348 model.phrase.xml_alternation = att | gi | val
349 model.phrase.xml_sequence = att, gi, val
350 model.phrase.xml_sequenceOptional = att?, gi?, val?
351 model.phrase.xml_sequenceOptionalRepeatable = att*, gi*, val*
352 model.phrase.xml_sequenceRepeatable = att+, gi+, val+
353 model.biblLike = bibl | biblStruct
354 model.biblLike_alternation = bibl | biblStruct
355 model.biblLike_sequence = bibl, biblStruct
356 model.biblLike_sequenceOptional = bibl?, biblStruct?
357 model.biblLike_sequenceOptionalRepeatable = bibl*, biblStruct*
358 model.biblLike_sequenceRepeatable = bibl+, biblStruct+
359 model.headLike = notAllowed
360 model.headLike_alternation = notAllowed
361 model.headLike_sequence = empty
362 model.headLike_sequenceOptional = empty
363 model.headLike_sequenceOptionalRepeatable = empty
364 model.headLike_sequenceRepeatable = notAllowed
365 model.listLike = \list | listBibl | listPerson
366 model.listLike_alternation = \list | listBibl | listPerson
367 model.listLike_sequence = \list, listBibl, listPerson
368 model.listLike_sequenceOptional = \list?, listBibl?, listPerson?
369 model.listLike_sequenceOptionalRepeatable =
370   \list*, listBibl*, listPerson*
371 model.listLike_sequenceRepeatable = \list+, listBibl+, listPerson+
372 model.noteLike = note
373 model.noteLike_alternation = note
374 model.noteLike_sequence = note
375 model.noteLike_sequenceOptional = note?
376 model.noteLike_sequenceOptionalRepeatable = note*
377 model.noteLike_sequenceRepeatable = note+
378 model.pLike = p
379 model.pLike_alternation = p
380 model.pLike_sequence = p
381 model.pLike_sequenceOptional = p?
382 model.pLike_sequenceOptionalRepeatable = p*
383 model.pLike_sequenceRepeatable = p+
384 model.global.edit = gap
385 model.global.edit_alternation = gap
386 model.global.edit_sequence = gap
387 model.global.edit_sequenceOptional = gap?
388 model.global.edit_sequenceOptionalRepeatable = gap*
389 model.global.edit_sequenceRepeatable = gap+
390 model.divPart = model.pLike | model.divPart.spoken
391 model.divPart_alternation =
392   model.pLike_alternation | model.divPart.spoken_alternation
393 model.divPart_sequence =
394   model.pLike_sequence, model.divPart.spoken_sequence
395 model.divPart_sequenceOptional =
396   model.pLike_sequenceOptional?, model.divPart.spoken_sequenceOptional?
397 model.divPart_sequenceOptionalRepeatable =
398   model.pLike_sequenceOptionalRepeatable*,
399   model.divPart.spoken_sequenceOptionalRepeatable*
400 model.divPart_sequenceRepeatable =
401   model.pLike_sequenceRepeatable+,
402   model.divPart.spoken_sequenceRepeatable+
403 model.persTraitLike = age | langKnowledge | sex
404 model.persTraitLike_alternation = age | langKnowledge | sex
405 model.persTraitLike_sequence = age, langKnowledge, sex
406 model.persTraitLike_sequenceOptional = age?, langKnowledge?, sex?
407 model.persTraitLike_sequenceOptionalRepeatable =
```

```
408   age*, langKnowledge*, sex*
409 model.persTraitLike_sequenceRepeatable = age+, langKnowledge+, sex+
410 model.persStateLike = persName | occupation
411 model.persStateLike_alternation = persName | occupation
412 model.persStateLike_sequence = persName, occupation
413 model.persStateLike_sequenceOptional = persName?, occupation?
414 model.persStateLike_sequenceOptionalRepeatable = persName*, occupation*
415 model.persStateLike_sequenceRepeatable = persName+, occupation+
416 model.personLike = person | personGrp
417 model.personPart = model.persTraitLike | model.persStateLike | bibl
418 model.publicationStmtPart =
419   address
420   | date
421   | publisher
422   | pubPlace
423   | distributor
424   | idno
425   | availability
426 model.glossLike = notAllowed
427 model.respLike =
428   author | editor | respStmt | sponsor | funder | principal
429 model.respLike_alternation =
430   author | editor | respStmt | sponsor | funder | principal
431 model.respLike_sequence =
432   author, editor, respStmt, sponsor, funder, principal
433 model.respLike_sequenceOptional =
434   author?, editor?, respStmt?, sponsor?, funder?, principal?
435 model.respLike_sequenceOptionalRepeatable =
436   author*, editor*, respStmt*, sponsor*, funder*, principal*
437 model.respLike_sequenceRepeatable =
438   author+, editor+, respStmt+, sponsor+, funder+, principal+
439 model.divTopPart = model.headLike
440 model.divTopPart_alternation = model.headLike_alternation
441 model.divTopPart_sequence = model.headLike_sequence
442 model.divTopPart_sequenceOptional = model.headLike_sequenceOptional?
443 model.divTopPart_sequenceOptionalRepeatable =
444   model.headLike_sequenceOptionalRepeatable*
445 model.divTopPart_sequenceRepeatable = model.headLike_sequenceRepeatable+
446 model.divTop = model.divTopPart
447 model.divBottom = notAllowed
448 model.msQuoteLike = title
449 model.msQuoteLike_alternation = title
450 model.msQuoteLike_sequence = title
451 model.msQuoteLike_sequenceOptional = title?
452 model.msQuoteLike_sequenceOptionalRepeatable = title*
453 model.msQuoteLike_sequenceRepeatable = title+
454 model.imprintPart = publisher | biblScope | pubPlace | distributor
455 model.imprintPart_alternation =
456   publisher | biblScope | pubPlace | distributor
457 model.imprintPart_sequence = publisher, biblScope, pubPlace, distributor
458 model.imprintPart_sequenceOptional =
459   publisher?, biblScope?, pubPlace?, distributor?
460 model.imprintPart_sequenceOptionalRepeatable =
461   publisher*, biblScope*, pubPlace*, distributor*
462 model.imprintPart_sequenceRepeatable =
463   publisher+, biblScope+, pubPlace+, distributor+
464 model.catDescPart = notAllowed
465 model.settingPart = locale | activity
466 model.addressLike = address
467 model.addressLike_alternation = address
468 model.addressLike_sequence = address
469 model.addressLike_sequenceOptional = address?
470 model.addressLike_sequenceOptionalRepeatable = address*
```

```
model.addressLike_sequenceRepeatable = address+
model.nameLike = model.nameLike.agent | idno
model.nameLike_alternation = model.nameLike.agent_alternation | idno
model.nameLike_sequence = model.nameLike.agent_sequence , idno
model.nameLike_sequenceOptional =
  model.nameLike.agent_sequenceOptional?, idno?
model.nameLike_sequenceOptionalRepeatable =
  model.nameLike.agent_sequenceOptionalRepeatable*, idno*
model.nameLike_sequenceRepeatable =
  model.nameLike.agent_sequenceRepeatable+, idno+
model.global =
  model.milestoneLike
  | model.noteLike
  | model.global.edit
  | model.global.spoken
  | track
model.biblPart = model.respLike | model.imprintPart | edition | extent
model.addrPart = model.nameLike | addrLine
model.pPart.data = model.dateLike | model.addressLike | model.nameLike
model.pPart.data_alternation =
  model.dateLike_alternation
  | model.addressLike_alternation
  | model.nameLike_alternation
model.pPart.data_sequence =
  model.dateLike_sequence ,
  model.addressLike_sequence ,
  model.nameLike_sequence
model.pPart.data_sequenceOptional =
  model.dateLike_sequenceOptional?,
  model.addressLike_sequenceOptional?,
  model.nameLike_sequenceOptional?
model.pPart.data_sequenceOptionalRepeatable =
  model.dateLike_sequenceOptionalRepeatable*,
  model.addressLike_sequenceOptionalRepeatable*,
  model.nameLike_sequenceOptionalRepeatable*
model.pPart.data_sequenceRepeatable =
  model.dateLike_sequenceRepeatable+,
  model.addressLike_sequenceRepeatable+,
  model.nameLike_sequenceRepeatable+
model.inter = model.oddRef | model.biblLike | model.listLike
model.inter_alternation =
  model.oddRef_alternation
  | model.biblLike_alternation
  | model.listLike_alternation
model.inter_sequence =
  model.oddRef_sequence ,
  model.biblLike_sequence ,
  model.listLike_sequence
model.inter_sequenceOptional =
  model.oddRef_sequenceOptional?,
  model.biblLike_sequenceOptional?,
  model.listLike_sequenceOptional?
model.inter_sequenceOptionalRepeatable =
  model.oddRef_sequenceOptionalRepeatable*,
  model.biblLike_sequenceOptionalRepeatable*,
  model.listLike_sequenceOptionalRepeatable*
model.inter_sequenceRepeatable =
  model.oddRef_sequenceRepeatable+,
  model.biblLike_sequenceRepeatable+,
  model.listLike_sequenceRepeatable+
model.common = model.divPart | model.inter
model.phrase =
  model.segLike
```

```
534     | model.highlighted
535     | model.pPart.edit
536     | model.ptrLike
537     | model.lPart
538     | model.phrase.xml
539     | model.pPart.data
540     | pvc
541   model.limitedPhrase =
542     model.emphLike | model.ptrLike | model.phrase.xml | model.pPart.data
543   model.divLike = \div
544   model.divGenLike = notAllowed
545   model.div1Like = notAllowed
546   model.teiHeaderPart = encodingDesc | profileDesc
547   model.sourceDescPart = scriptStmt | recordingStmt
548   model.encodingDescPart =
549     projectDesc | samplingDecl | editorialDecl | tagsDecl | classDecl
550   model.editorialDeclPart = correction | normalization | segmentation
551   model.profileDescPart =
552     particDesc | settingDesc | creation | langUsage | textClass
553   model.resourceLike = notAllowed
554   att.personal.attributes = att.naming.attributes
555   model.placeLike = notAllowed
556   p =
557
558     ## (paragraph) marks paragraphs in prose.
559     element tei:p {
560       macro.paraContent,
561       att.global.attribute.xmlid,
562       att.global.attribute.xmlspace,
563       att.global.linking.attributes,
564       att.global.analytic.attributes,
565       att.global.facs.attributes,
566       att.declaring.attributes,
567       empty
568     }
569   foreign =
570
571     ## (foreign) identifies a word or phrase as belonging to some
572     ## language other than that of the surrounding text.
573     element tei:foreign {
574       macro.phraseSeq,
575
576       ## Translation or comment in English regarding the meaning of the
577       ## element's content
578       attribute voice:translation { text }?,
579       att.global.attribute.xmlid,
580       att.global.attribute.xmllang,
581       att.global.attribute.xmlspace,
582       att.global.linking.attributes,
583       att.global.analytic.attributes,
584       att.global.facs.attributes,
585       att.typed.attributes,
586       empty
587     }
588   emph =
589
590     ## (emphasized) marks words or phrases which are stressed or
591     ## emphasized for linguistic or rhetorical effect.
592     element tei:emph {
593       macro.paraContent,
594       att.global.attribute.xmlid,
595       att.global.attribute.xmlspace,
596       att.global.linking.attributes,
```

```
597       att.global.analytic.attributes,
598       att.global.facs.attributes,
599       empty
600     }
601   gap =
602
603     ## (gap) indicates a point where material has been omitted in a
604     ## transcription, whether for editorial reasons described in the TEI
605     ## header, as part of sampling practice, or because the material is
606     ## illegible, invisible, or inaudible.
607     element tei:gap {
608       model.glossLike*,
609
610       attribute voice:desc { text }?,
611
612       ## gives the reason for omission. Sample values include sampling,
613       ## inaudible, irrelevant, cancelled.
614       attribute reason {
615         list {
616           xsd:token { pattern = "(\p{L}|\p{N}|\p{P}|\p{S})+" },
617           xsd:token { pattern = "(\p{L}|\p{N}|\p{P}|\p{S})+" }*
618         }
619       }?,
620       att.global.attribute.xmlid,
621       att.global.attribute.xmlspace,
622       att.global.linking.attributes,
623       att.global.analytic.attributes,
624       att.global.facs.attributes,
625       att.duration.w3c.attributes,
626       att.duration.iso.attributes,
627       att.editLike.attributes,
628       empty
629     }
630   unclear =
631
632     ## contains a word, phrase, or passage which cannot be transcribed
633     ## with certainty because it is illegible or inaudible in the
634     ## source.
635     element tei:unclear {
636       macro.paraContent,
637       att.global.attribute.xmlid,
638       att.global.attribute.xmlspace,
639       att.global.linking.attributes,
640       att.global.analytic.attributes,
641       att.global.facs.attributes,
642       att.editLike.attributes,
643       empty
644     }
645   name =
646
647     ## (name, proper noun) contains a proper noun or noun phrase.
648     element tei:name {
649       macro.phraseSeq,
650       att.global.attribute.xmlid,
651       att.global.attribute.xmlspace,
652       att.global.linking.attributes,
653       att.global.analytic.attributes,
654       att.global.facs.attributes,
655       att.naming.attributes,
656       att.typed.attribute.type,
657       empty
658     }
659   address =
```

```
660
661   ## contains a postal  address, for example of a
662   ## publisher, an organization, or an individual.
663   element tei:address {
664     (model.global*, (model.addrPart, model.global*)+),
665     att.global.attribute.xmlid,
666     att.global.attribute.xmlspace,
667     att.global.linking.attributes,
668     att.global.analytic.attributes,
669     att.global.facs.attributes,
670     empty
671   }
672 addrLine =
673
674   ## (address line) contains one line of a postal address.
675   element tei:addrLine {
676     macro.phraseSeq,
677     att.global.attribute.xmlid,
678     att.global.attribute.xmlspace,
679     att.global.linking.attributes,
680     att.global.analytic.attributes,
681     att.global.facs.attributes,
682     empty
683   }
684 date =
685
686   ## contains a date in any format.
687   element tei:date {
688     (text | model.gLike | model.phrase | model.global)*,
689     att.global.attribute.xmlid,
690     att.global.attribute.xmlspace,
691     att.global.linking.attributes,
692     att.global.analytic.attributes,
693     att.global.facs.attributes,
694     att.datable.w3c.attributes,
695     att.datable.iso.attributes,
696     att.duration.iso.attributes,
697     att.editLike.attributes,
698     empty
699   }
700 ref =
701
702   ## (reference) defines a reference to another location, possibly
703   ## modified by additional text or comment.
704   element tei:ref {
705     macro.paraContent
706     >> sch:pattern [
707          id = "ref-constraint-refAtts"
708          "\x{a}" ~
709          "␣␣␣␣␣␣␣␣␣␣␣␣"
710          sch:rule [
711            context = "tei:ref"
712            "\x{a}" ~
713            "␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣"
714            sch:report [
715              test = "@target␣and␣@cRef"
716              "Only␣one␣of␣the\x{a}" ~
717              "␣␣attributes␣'target'␣and␣'cRef'␣may␣be␣supplied."
718            ]
719            "\x{a}" ~
720            "␣␣␣␣␣␣␣␣␣␣␣␣"
721          ]
722          "\x{a}" ~
```

```
723             "␣␣␣␣␣␣␣␣␣"
724           ],
725       att.global.attribute.xmlid,
726       att.global.attribute.xmlspace,
727       att.global.linking.attributes,
728       att.global.analytic.attributes,
729       att.global.facs.attributes,
730       att.pointing.attributes,
731       att.typed.attributes,
732       att.declaring.attributes,
733       empty
734     }
735   \list =
736
737     ## (list) contains any sequence of items organized as a list.
738     element tei:list {
739       ((model.divTop | model.global)*,
740        ((item, model.global*)+ | (model.global*, item, model.global*)+),
741        (model.divBottom, model.global*)*),
742
743       ## describes the form of the list.
744       [ a:defaultValue = "simple" ]
745       attribute type {
746
747         "ordered"
748       }?,
749       att.global.attribute.xmlid,
750
751       ## (rendition) indicates how the element in question was rendered
752       ## or presented in the source text.
753       attribute rend {
754         list {
755           (
756            "p"),
757           (
758            "p")*
759         }
760       }?,
761       att.global.attribute.xmlspace,
762       att.global.linking.attributes,
763       att.global.analytic.attributes,
764       att.global.facs.attributes,
765       empty
766     }
767   item =
768
769     ## contains one component of a list.
770     element tei:item {
771       macro.specialPara,
772       att.global.attribute.xmlid,
773       att.global.attribute.xmlspace,
774       att.global.linking.attributes,
775       att.global.analytic.attributes,
776       att.global.facs.attributes,
777       empty
778     }
779   note =
780
781     ## contains a note or annotation.
782     element tei:note {
783       macro.specialPara,
784       att.global.attribute.xmlid,
785       att.global.attribute.xmlspace,
```

```
786       att.global.linking.attributes ,
787       att.global.analytic.attributes ,
788       att.global.facs.attributes ,
789       att.placement.attributes ,
790       att.responsibility.attribute.cert ,
791       att.typed.attribute.subtype ,
792       empty
793     }
794   analytic =
795
796     ## (analytic level) contains bibliographic elements describing an
797     ## item (e.g. an article or poem) published within a monograph or
798     ## journal and not as an independent publication.
799     element tei:analytic {
800       (author | editor | respStmt | title | ref)*,
801       att.global.attribute.xmlid ,
802       att.global.attribute.xmlspace ,
803       att.global.linking.attributes ,
804       att.global.analytic.attributes ,
805       att.global.facs.attributes ,
806       empty
807     }
808   monogr =
809
810     ## (monographic level) contains bibliographic elements describing an
811     ## item (e.g. a book or journal) published as an independent item
812     ## (i.e. as a separate physical object).
813     element tei:monogr {
814       ((((author | editor | respStmt),
815          (author | editor | respStmt)*,
816          title+,
817          (idno | editor | respStmt)*)
818         | ((title | ref)+, (idno | author | editor | respStmt)*))?,
819        model.noteLike*,
820        (edition, (idno | editor | respStmt)*)*,
821        imprint,
822        (imprint | extent | biblScope)*),
823       att.global.attribute.xmlid ,
824       att.global.attribute.xmlspace ,
825       att.global.linking.attributes ,
826       att.global.analytic.attributes ,
827       att.global.facs.attributes ,
828       empty
829     }
830   author =
831
832     ## in a bibliographic reference, contains the name(s) of the
833     ## author(s), personal or corporate, of a work; for example in the
834     ## same form as that provided by a recognized bibliographic name
835     ## authority.
836     element tei:author {
837       macro.phraseSeq ,
838       att.global.attribute.xmlid ,
839       att.global.attribute.xmlspace ,
840       att.global.linking.attributes ,
841       att.global.analytic.attributes ,
842       att.global.facs.attributes ,
843       att.naming.attributes ,
844       empty
845     }
846   editor =
847
848     ## secondary statement of responsibility for a bibliographic item,
```

```
849   ## for example the name of an individual, institution or
850   ## organization, (or of several such) acting as editor, compiler,
851   ## translator, etc.
852   element tei:editor {
853     macro.phraseSeq,
854     att.global.attribute.xmlid,
855     att.global.attribute.xmlspace,
856     att.global.linking.attributes,
857     att.global.analytic.attributes,
858     att.global.facs.attributes,
859     att.canonical.attributes,
860     empty
861   }
862 respStmt =
863
864   ## (statement of responsibility) supplies a statement of
865   ## responsibility for the intellectual content of a text, edition,
866   ## recording, or series, where the specialized elements for authors,
867   ## editors, etc. do not suffice or do not apply.
868   element tei:respStmt {
869     ((resp+, model.nameLike.agent+) | (model.nameLike.agent+, resp+)),
870     att.global.attribute.xmlid,
871     att.global.attribute.xmlspace,
872     att.global.linking.attributes,
873     att.global.analytic.attributes,
874     att.global.facs.attributes,
875     empty
876   }
877 resp =
878
879   ## (responsibility) contains a phrase describing the nature of a
880   ## person's intellectual responsibility.
881   element tei:resp {
882     macro.phraseSeq.limited,
883     att.global.attribute.xmlid,
884     att.global.attribute.xmlspace,
885     att.global.linking.attributes,
886     att.global.analytic.attributes,
887     att.global.facs.attributes,
888     att.canonical.attributes,
889     empty
890   }
891 title =
892
893   ## contains a title for any kind of work.
894   element tei:title {
895     macro.paraContent,
896
897     ## indicates the bibliographic level for a title, that is, whether
898     ## it identifies an article, book, journal, series, or unpublished
899     ## material.
900     attribute level {
901
902       "j"
903       |
904         "m"
905     }?,
906     att.global.attribute.xmlid,
907     att.global.attribute.xmlspace,
908     att.global.linking.attributes,
909     att.global.analytic.attributes,
910     att.global.facs.attributes,
911     att.canonical.attributes,
```

```
912       empty
913     }
914   imprint =
915
916     ## groups information relating to the publication or distribution of
917     ## a bibliographic item.
918     element tei:imprint {
919       ((model.imprintPart | model.dateLike), model.global*)+,
920       att.global.attribute.xmlid,
921       att.global.attribute.xmlspace,
922       att.global.linking.attributes,
923       att.global.analytic.attributes,
924       att.global.facs.attributes,
925       empty
926     }
927   publisher =
928
929     ## provides the name of the organization responsible for the
930     ## publication or distribution of a bibliographic item.
931     element tei:publisher {
932       macro.phraseSeq,
933       att.global.attribute.xmlid,
934       att.global.attribute.xmlspace,
935       att.global.linking.attributes,
936       att.global.analytic.attributes,
937       att.global.facs.attributes,
938       empty
939     }
940   biblScope =
941
942     ## (scope of citation) defines the scope of a bibliographic
943     ## reference, for example as a list of page numbers, or a named
944     ## subdivision of a larger work.
945     element tei:biblScope {
946       macro.phraseSeq,
947
948       ## identifies the type of information conveyed by the element,
949       ## e.g. columns, pages, volume.
950       attribute type {
951
952         "issue"
953         |
954           "pp"
955         |
956           "vol"
957       }?,
958
959       ## specifies the starting point of the range of units indicated by
960       ## the type attribute.
961       attribute from {
962         xsd:token { pattern = "(\p{L}|\p{N}|\p{P}|\p{S})+" }
963       }?,
964
965       ## specifies the end-point of the range of units indicated by the
966       ## type attribute.
967       attribute to {
968         xsd:token { pattern = "(\p{L}|\p{N}|\p{P}|\p{S})+" }
969       }?,
970       att.global.attribute.xmlid,
971       att.global.attribute.xmlspace,
972       att.global.linking.attributes,
973       att.global.analytic.attributes,
974       att.global.facs.attributes,
```

```
975       empty
976     }
977   pubPlace =
978
979     ## (publication place) contains the name of the place where a
980     ## bibliographic item was published.
981     element tei:pubPlace {
982       macro.phraseSeq,
983       att.global.attribute.xmlid,
984       att.global.attribute.xmlspace,
985       att.global.linking.attributes,
986       att.global.analytic.attributes,
987       att.global.facs.attributes,
988       att.naming.attributes,
989       empty
990     }
991   bibl =
992
993     ## (bibliographic citation) contains a loosely-structured
994     ## bibliographic citation of which the sub-components may or may not
995     ## be explicitly tagged.
996     element tei:bibl {
997       (text
998        | model.gLike
999        | model.highlighted
1000       | model.pPart.data
1001       | model.pPart.edit
1002       | model.segLike
1003       | model.ptrLike
1004       | model.biblPart
1005       | model.global)*,
1006      att.global.attribute.xmlid,
1007      att.global.attribute.xmlspace,
1008      att.global.linking.attributes,
1009      att.global.analytic.attributes,
1010      att.global.facs.attributes,
1011      att.declarable.attributes,
1012      empty
1013    }
1014  biblStruct =
1015
1016    ## (structured bibliographic citation) contains a structured
1017    ## bibliographic citation, in which only bibliographic sub-elements
1018    ## appear and in a specified order.
1019    element tei:biblStruct {
1020      (analytic*, monogr+, (model.noteLike | idno)*),
1021      att.global.attribute.xmlid,
1022      att.global.attribute.xmlspace,
1023      att.global.linking.attributes,
1024      att.global.analytic.attributes,
1025      att.global.facs.attributes,
1026      att.declarable.attributes,
1027      empty
1028    }
1029  listBibl =
1030
1031    ## (citation list) contains a list of bibliographic citations of any
1032    ## kind.
1033    element tei:listBibl {
1034      (model.headLike*,
1035       (model.biblLike | model.milestoneLike | listBibl)+),
1036      att.global.attribute.xmlid,
1037      att.global.attribute.xmlspace,
```

```
1038      att.global.linking.attributes,
1039      att.global.analytic.attributes,
1040      att.global.facs.attributes,
1041      att.declarable.attributes,
1042      empty
1043    }
1044  teiCorpus =
1045
1046    ## contains the whole of a TEI encoded corpus, comprising a single
1047    ## corpus header and one or more TEI elements, each containing a
1048    ## single text header and a text.
1049    element tei:teiCorpus {
1050      (teiHeader, (TEI | teiCorpus)+),
1051
1052      ## The version of the TEI scheme
1053      [ a:defaultValue = "5.0" ]
1054      attribute version {
1055        xsd:token { pattern = "[\d]+(\.[\d]+){0,2}" }
1056      }?,
1057      att.global.attribute.xmlid,
1058      att.global.attribute.xmlspace,
1059      att.global.linking.attributes,
1060      att.global.analytic.attributes,
1061      att.global.facs.attributes,
1062      empty
1063    }
1064  w =
1065
1066    ## (word) represents a grammatical (not necessarily orthographic)
1067    ## word.
1068    element tei:w {
1069      (text
1070       | model.gLike
1071       | seg
1072       | w
1073       | c
1074       | model.global
1075       | model.lPart
1076       | model.hiLike
1077       | model.pPart.edit)*,
1078
1079      ## Describes the mode a word is uttered in
1080      attribute voice:mode { text }?,
1081      att.global.attribute.xmlid,
1082      att.global.attribute.xmlspace,
1083      att.global.linking.attributes,
1084      att.global.analytic.attributes,
1085      att.global.facs.attributes,
1086      empty
1087    }
1088  c =
1089
1090    ## (character) represents a character.
1091    element tei:c {
1092      macro.xtext,
1093      att.global.attribute.xmlid,
1094      att.global.attribute.xmlspace,
1095      att.global.linking.attributes,
1096      att.global.analytic.attributes,
1097      att.global.facs.attributes,
1098      att.segLike.attributes,
1099      att.typed.attribute.type,
1100      empty
```

```
1101    }
1102  pc =
1103
1104    ## (punctuation character) a character or string of characters
1105    ## regarded as constituting a single punctuation mark.
1106    element tei:pc {
1107      (text | model.gLike | c)*,
1108      att.global.attributes,
1109      att.segLike.attributes,
1110      att.typed.attributes,
1111
1112      ## indicates the extent to which this punctuation mark
1113      ## conventionally separates words or phrases
1114      attribute force {
1115
1116        ## the punctuation mark is a word separator
1117        "strong"
1118        |
1119          ## the punctuation mark is not a word separator
1120          "weak"
1121        |
1122          ## the punctuation mark may or may not be a word separator
1123          "inter"
1124      }?,
1125
1126      ## provides a name for the kind of unit delimited by this
1127      ## punctuation mark.
1128      attribute unit { xsd:Name }?,
1129
1130      ## indicates whether this punctuation mark precedes or follows the
1131      ## unit it delimits.
1132      attribute pre { xsd:boolean }?,
1133      empty
1134    }
1135  att.global.analytic.attributes = empty
1136  particDesc =
1137
1138    ## (participation description) describes the identifiable speakers,
1139    ## voices, or other participants in any kind of text.
1140    element tei:particDesc {
1141      (model.pLike+ | (model.personLike | listPerson)+),
1142      att.global.attribute.xmlid,
1143      att.global.attribute.xmlspace,
1144      att.global.linking.attributes,
1145      att.global.analytic.attributes,
1146      att.global.facs.attributes,
1147      att.declarable.attributes,
1148      empty
1149    }
1150  settingDesc =
1151
1152    ## (setting description) describes the setting or settings within
1153    ## which a language interaction takes place, either as a prose
1154    ## description or as a series of setting elements.
1155    element tei:settingDesc {
1156      (model.pLike+ | (setting | model.placeLike)+),
1157      att.global.attribute.xmlid,
1158      att.global.attribute.xmlspace,
1159      att.global.linking.attributes,
1160      att.global.analytic.attributes,
1161      att.global.facs.attributes,
1162      att.declarable.attributes,
1163      empty
```

```
1164    }
1165  setting =
1166
1167    ## describes one particular setting in which a language interaction
1168    ## takes place.
1169    element tei:setting {
1170      (model.pLike+
1171       | (model.nameLike.agent | model.dateLike | model.settingPart)*),
1172      att.global.attribute.xmlid,
1173      att.global.attribute.xmlspace,
1174      att.global.linking.attributes,
1175      att.global.analytic.attributes,
1176      att.global.facs.attributes,
1177      empty
1178    }
1179  locale =
1180
1181    ## contains a brief informal description of the kind of place
1182    ## concerned, for example: a room, a restaurant, a park bench, etc.
1183    element tei:locale {
1184      macro.phraseSeq.limited,
1185      att.global.attribute.xmlid,
1186      att.global.attribute.xmlspace,
1187      att.global.linking.attributes,
1188      att.global.analytic.attributes,
1189      att.global.facs.attributes,
1190      empty
1191    }
1192  activity =
1193
1194    ## contains a brief informal description of what a participant in a
1195    ## language interaction is doing other than speaking, if anything.
1196    element tei:activity {
1197      macro.phraseSeq.limited,
1198      att.global.attribute.xmlid,
1199      att.global.attribute.xmlspace,
1200      att.global.linking.attributes,
1201      att.global.analytic.attributes,
1202      att.global.facs.attributes,
1203      empty
1204    }
1205  teiHeader =
1206
1207    ## (TEI Header) supplies the descriptive and declarative information
1208    ## making up an electronic title page prefixed to every
1209    ## TEI-conformant text.
1210    element tei:teiHeader {
1211      (fileDesc, model.teiHeaderPart*, revisionDesc?),
1212
1213      ## specifies the kind of document to which the header is attached,
1214      ## for example whether it is a corpus or individual text.
1215      [ a:defaultValue = "text" ]
1216      attribute type {
1217
1218        "corpus"
1219      }?,
1220      att.global.attribute.xmlid,
1221      att.global.attribute.xmlspace,
1222      att.global.linking.attributes,
1223      att.global.analytic.attributes,
1224      att.global.facs.attributes,
1225      empty
1226    }
```

```
fileDesc =

  ## (file description) contains a full bibliographic description of
  ## an electronic file.
  element tei:fileDesc {
    ((titleStmt, editionStmt?, extent?, publicationStmt, notesStmt?),
     sourceDesc+),
    att.global.attribute.xmlid,
    att.global.attribute.xmlspace,
    att.global.linking.attributes,
    att.global.analytic.attributes,
    att.global.facs.attributes,
    empty
  }
titleStmt =

  ## (title statement) groups information about the title of a work
  ## and those responsible for its intellectual content.
  element tei:titleStmt {
    (title+, model.respLike*),
    att.global.attribute.xmlid,
    att.global.attribute.xmlspace,
    att.global.linking.attributes,
    att.global.analytic.attributes,
    att.global.facs.attributes,
    empty
  }
sponsor =

  ## specifies the name of a sponsoring organization or institution.
  element tei:sponsor {
    macro.phraseSeq.limited,
    att.global.attribute.xmlid,
    att.global.attribute.xmlspace,
    att.global.linking.attributes,
    att.global.analytic.attributes,
    att.global.facs.attributes,
    empty
  }
funder =

  ## (funding body) specifies the name of an individual, institution,
  ## or organization responsible for the funding of a project or text.
  element tei:funder {
    macro.phraseSeq.limited,
    att.global.attribute.xmlid,
    att.global.attribute.xmlspace,
    att.global.linking.attributes,
    att.global.analytic.attributes,
    att.global.facs.attributes,
    empty
  }
principal =

  ## (principal researcher) supplies the name of the principal
  ## researcher responsible for the creation of an electronic text.
  element tei:principal {
    macro.phraseSeq.limited,
    att.global.attribute.xmlid,
    att.global.attribute.xmlspace,
    att.global.linking.attributes,
    att.global.analytic.attributes,
    att.global.facs.attributes,
```

```
1290      empty
1291    }
1292  editionStmt =
1293
1294    ## (edition statement) groups information relating to one edition of
1295    ## a text.
1296    element tei:editionStmt {
1297      (model.pLike+ | (edition, respStmt*)),
1298      att.global.attribute.xmlid,
1299      att.global.attribute.xmlspace,
1300      att.global.linking.attributes,
1301      att.global.analytic.attributes,
1302      att.global.facs.attributes,
1303      empty
1304    }
1305  edition =
1306
1307    ## (edition) describes the particularities of one edition of a text.
1308    element tei:edition {
1309      macro.phraseSeq,
1310      att.global.attribute.xmlid,
1311      att.global.attribute.xmlspace,
1312      att.global.linking.attributes,
1313      att.global.analytic.attributes,
1314      att.global.facs.attributes,
1315      empty
1316    }
1317  extent =
1318
1319    ## describes the approximate size of a text as stored on some
1320    ## carrier medium, whether digital or non-digital, specified in any
1321    ## convenient units.
1322    element tei:extent {
1323      macro.phraseSeq,
1324      att.global.attribute.xmlid,
1325      att.global.attribute.xmlspace,
1326      att.global.linking.attributes,
1327      att.global.analytic.attributes,
1328      att.global.facs.attributes,
1329      empty
1330    }
1331  publicationStmt =
1332
1333    ## (publication statement) groups information concerning the
1334    ## publication or distribution of an electronic or other text.
1335    element tei:publicationStmt {
1336      (model.pLike+ | model.publicationStmtPart+),
1337      att.global.attribute.xmlid,
1338      att.global.attribute.xmlspace,
1339      att.global.linking.attributes,
1340      att.global.analytic.attributes,
1341      att.global.facs.attributes,
1342      empty
1343    }
1344  distributor =
1345
1346    ## supplies the name of a person or other agency responsible for the
1347    ## distribution of a text.
1348    element tei:distributor {
1349      macro.phraseSeq,
1350      att.global.attribute.xmlid,
1351      att.global.attribute.xmlspace,
1352      att.global.linking.attributes,
```

```
1353      att.global.analytic.attributes ,
1354      att.global.facs.attributes ,
1355      empty
1356    }
1357  idno =
1358
1359    ## (identifier) supplies any form of identifier used to identify
1360    ## some object, such as a bibliographic item, a person, a title, an
1361    ## organization, etc. in a standardized way.
1362    element tei:idno {
1363      text ,
1364      att.global.attribute.xmlid ,
1365      att.global.attribute.xmlspace ,
1366      att.global.linking.attributes ,
1367      att.global.analytic.attributes ,
1368      att.global.facs.attributes ,
1369      empty
1370    }
1371  availability =
1372
1373    ## supplies information about the availability of a text, for
1374    ## example any restrictions on its use or distribution, its
1375    ## copyright status, etc.
1376    element tei:availability {
1377      model.pLike+,
1378
1379      ## supplies a code identifying the current availability of the
1380      ## text.
1381      [ a:defaultValue = "unknown" ]
1382      attribute status {
1383
1384        "free"
1385      }?,
1386      att.global.attribute.xmlid ,
1387      att.global.attribute.xmlbase ,
1388      att.global.attribute.xmlspace ,
1389      att.global.linking.attributes ,
1390      att.global.analytic.attributes ,
1391      att.global.facs.attributes ,
1392      att.declarable.attributes ,
1393      empty
1394    }
1395  notesStmt =
1396
1397    ## (notes statement) collects together any notes providing
1398    ## information about a text additional to that recorded in other
1399    ## parts of the bibliographic description.
1400    element tei:notesStmt {
1401      model.noteLike+,
1402      att.global.attribute.xmlid ,
1403      att.global.attribute.xmlspace ,
1404      att.global.linking.attributes ,
1405      att.global.analytic.attributes ,
1406      att.global.facs.attributes ,
1407      empty
1408    }
1409  sourceDesc =
1410
1411    ## (source description) describes the source from which an
1412    ## electronic text was derived or generated, typically a
1413    ## bibliographic description in the case of a digitized text, or a
1414    ## phrase such as "born digital" for a text which has no previous
1415    ## existence.
```

```
1416    element tei:sourceDesc {
1417      (model.pLike+
1418       | (model.biblLike | model.sourceDescPart | model.listLike)+),
1419      att.global.attribute.xmlid ,
1420      att.global.attribute.xmlspace ,
1421      att.global.linking.attributes ,
1422      att.global.analytic.attributes ,
1423      att.global.facs.attributes ,
1424      att.declarable.attributes ,
1425      empty
1426    }
1427  encodingDesc =
1428
1429    ## (encoding description) documents the relationship between an
1430    ## electronic text and the source or sources from which it was
1431    ## derived.
1432    element tei:encodingDesc {
1433      (model.encodingDescPart | model.pLike)+,
1434      att.global.attribute.xmlid ,
1435      att.global.attribute.xmlspace ,
1436      att.global.linking.attributes ,
1437      att.global.analytic.attributes ,
1438      att.global.facs.attributes ,
1439      empty
1440    }
1441  projectDesc =
1442
1443    ## (project description) describes in detail the aim or purpose for
1444    ## which an electronic file was encoded, together with any other
1445    ## relevant information concerning the process by which it was
1446    ## assembled or collected.
1447    element tei:projectDesc {
1448      model.pLike+,
1449      att.global.attribute.xmlid ,
1450      att.global.attribute.xmlspace ,
1451      att.global.linking.attributes ,
1452      att.global.analytic.attributes ,
1453      att.global.facs.attributes ,
1454      att.declarable.attributes ,
1455      empty
1456    }
1457  samplingDecl =
1458
1459    ## (sampling declaration) contains a prose description of the
1460    ## rationale and methods used in sampling texts in the creation of a
1461    ## corpus or collection.
1462    element tei:samplingDecl {
1463      model.pLike+,
1464      att.global.attribute.xmlid ,
1465      att.global.attribute.xmlspace ,
1466      att.global.linking.attributes ,
1467      att.global.analytic.attributes ,
1468      att.global.facs.attributes ,
1469      att.declarable.attributes ,
1470      empty
1471    }
1472  editorialDecl =
1473
1474    ## (editorial practice declaration) provides details of editorial
1475    ## principles and practices applied during the encoding of a text.
1476    element tei:editorialDecl {
1477      (model.pLike | model.editorialDeclPart)+,
1478      att.global.attribute.xmlid ,
```

```
1479      att.global.attribute.xmlspace,
1480      att.global.linking.attributes,
1481      att.global.analytic.attributes,
1482      att.global.facs.attributes,
1483      att.declarable.attributes,
1484      empty
1485    }
1486  correction =
1487
1488    ## (correction principles) states how and under what circumstances
1489    ## corrections have been made in the text.
1490    element tei:correction {
1491      model.pLike+,
1492
1493      ## indicates the degree of correction applied to the text.
1494      [ a:defaultValue = "unknown" ]
1495      attribute status {
1496
1497        "high"
1498      }?,
1499      att.global.attribute.xmlid,
1500      att.global.attribute.xmlspace,
1501      att.global.linking.attributes,
1502      att.global.analytic.attributes,
1503      att.global.facs.attributes,
1504      att.declarable.attributes,
1505      empty
1506    }
1507  normalization =
1508
1509    ## indicates the extent of normalization or regularization of the
1510    ## original source carried out in converting it to electronic form.
1511    element tei:normalization {
1512      model.pLike+,
1513      att.global.attribute.xmlid,
1514      att.global.attribute.xmlspace,
1515      att.global.linking.attributes,
1516      att.global.analytic.attributes,
1517      att.global.facs.attributes,
1518      att.declarable.attributes,
1519      empty
1520    }
1521  segmentation =
1522
1523    ## describes the principles according to which the text has been
1524    ## segmented, for example into sentences, tone-units, graphemic
1525    ## strata, etc.
1526    element tei:segmentation {
1527      model.pLike+,
1528      att.global.attribute.xmlid,
1529      att.global.attribute.xmlspace,
1530      att.global.linking.attributes,
1531      att.global.analytic.attributes,
1532      att.global.facs.attributes,
1533      att.declarable.attributes,
1534      empty
1535    }
1536  tagsDecl =
1537
1538    ## (tagging declaration) provides detailed information about the
1539    ## tagging applied to a document.
1540    element tei:tagsDecl {
1541      \namespace*,
```

```
1542      att.global.attribute.xmlid ,
1543      att.global.attribute.xmlspace ,
1544      att.global.linking.attributes ,
1545      att.global.analytic.attributes ,
1546      att.global.facs.attributes ,
1547      empty
1548    }
1549  tagUsage =
1550
1551    ## supplies information about the usage of a specific element within
1552    ## a text.
1553    element tei:tagUsage {
1554      macro.limitedContent ,
1555
1556      ## (element name) the name (generic identifier) of the element
1557      ## indicated by the tag.
1558      attribute gi { xsd:Name },
1559
1560      ## specifies the number of occurrences of this element within the
1561      ## text.
1562      attribute occurs { xsd:nonNegativeInteger }?,
1563      att.global.attribute.xmlid ,
1564      att.global.attribute.xmlspace ,
1565      att.global.linking.attributes ,
1566      att.global.analytic.attributes ,
1567      att.global.facs.attributes ,
1568      empty
1569    }
1570  \namespace =
1571
1572    ## supplies the formal name of the namespace to which the elements
1573    ## documented by its children belong.
1574    element tei:namespace {
1575      tagUsage+,
1576
1577      ## the full formal name of the namespace concerned.
1578      attribute name { xsd:anyURI },
1579      att.global.attribute.xmlid ,
1580      att.global.attribute.xmlspace ,
1581      att.global.linking.attributes ,
1582      att.global.analytic.attributes ,
1583      att.global.facs.attributes ,
1584      empty
1585    }
1586  classDecl =
1587
1588    ## (classification declarations) contains one or more taxonomies
1589    ## defining any classificatory codes used elsewhere in the text.
1590    element tei:classDecl {
1591      taxonomy+,
1592      att.global.attribute.xmlid ,
1593      att.global.attribute.xmlspace ,
1594      att.global.linking.attributes ,
1595      att.global.analytic.attributes ,
1596      att.global.facs.attributes ,
1597      empty
1598    }
1599  taxonomy =
1600
1601    ## defines a typology used to classify texts either implicitly, by
1602    ## means of a bibliographic citation, or explicitly by a structured
1603    ## taxonomy.
1604    element tei:taxonomy {
```

```
1605      (model.glossLike* | category+ | (model.biblLike, category*)),
1606      att.global.attribute.xmlid,
1607      att.global.attribute.xmlspace,
1608      att.global.linking.attributes,
1609      att.global.analytic.attributes,
1610      att.global.facs.attributes,
1611      empty
1612    }
1613  category =
1614
1615    ## contains an individual descriptive category, possibly nested
1616    ## within a superordinate category, within a user-defined taxonomy.
1617    element tei:category {
1618      ((catDesc+ | model.glossLike*), category*),
1619      att.global.attribute.xmlid,
1620      att.global.attribute.n,
1621      att.global.attribute.xmlspace,
1622      att.global.linking.attributes,
1623      att.global.analytic.attributes,
1624      att.global.facs.attributes,
1625      empty
1626    }
1627  catDesc =
1628
1629    ## (category description) describes some category within a taxonomy
1630    ## or text typology, either in the form of a brief prose description
1631    ## or in terms of the situational parameters used by the TEI formal
1632    ## textDesc.
1633    element tei:catDesc {
1634      (text | model.limitedPhrase | model.catDescPart)*,
1635      att.global.attribute.xmlid,
1636      att.global.attribute.xmlspace,
1637      att.global.linking.attributes,
1638      att.global.analytic.attributes,
1639      att.global.facs.attributes,
1640      empty
1641    }
1642  profileDesc =
1643
1644    ## (text-profile description) provides a detailed description of
1645    ## non-bibliographic aspects of a text, specifically the languages
1646    ## and sublanguages used, the situation in which it was produced,
1647    ## the participants and their setting.
1648    element tei:profileDesc {
1649      model.profileDescPart*,
1650      att.global.attribute.xmlid,
1651      att.global.attribute.xmlspace,
1652      att.global.linking.attributes,
1653      att.global.analytic.attributes,
1654      att.global.facs.attributes,
1655      empty
1656    }
1657  creation =
1658
1659    ## contains information about the creation of a text.
1660    element tei:creation {
1661      macro.phraseSeq.limited,
1662      att.global.attribute.xmlid,
1663      att.global.attribute.xmlspace,
1664      att.global.linking.attributes,
1665      att.global.analytic.attributes,
1666      att.global.facs.attributes,
1667      empty
```

```
1668    }
1669  langUsage =
1670
1671    ## (language usage) describes the languages, sublanguages,
1672    ## registers, dialects, etc.  represented within a text.
1673    element tei:langUsage {
1674      language+,
1675      att.global.attribute.xmlid,
1676      att.global.attribute.xmlspace,
1677      att.global.linking.attributes,
1678      att.global.analytic.attributes,
1679      att.global.facs.attributes,
1680      att.declarable.attributes,
1681      empty
1682    }
1683  language =
1684
1685    ## characterizes a single language or sublanguage used within a
1686    ## text.
1687    element tei:language {
1688      macro.phraseSeq.limited,
1689
1690      ## (identifier) Supplies a language code constructed as defined in
1691      ## BCP 47 which is used to identify the language documented by
1692      ## this element, and which is referenced by the global xml:lang
1693      ## attribute.
1694      attribute ident { xsd:language },
1695      att.global.attribute.xmlid,
1696      att.global.attribute.xmlspace,
1697      att.global.linking.attributes,
1698      att.global.analytic.attributes,
1699      att.global.facs.attributes,
1700      empty
1701    }
1702  textClass =
1703
1704    ## (text classification) groups information which describes the
1705    ## nature or topic of a text in terms of a standard classification
1706    ## scheme, thesaurus, etc.
1707    element tei:textClass {
1708      catRef*,
1709      att.global.attribute.xmlid,
1710      att.global.attribute.xmlspace,
1711      att.global.linking.attributes,
1712      att.global.analytic.attributes,
1713      att.global.facs.attributes,
1714      att.declarable.attributes,
1715      empty
1716    }
1717  catRef =
1718
1719    ## (category reference) specifies one or more defined categories
1720    ## within some taxonomy or text typology.
1721    element tei:catRef {
1722      empty,
1723      att.global.attribute.xmlid,
1724      att.global.attribute.xmlspace,
1725      att.global.linking.attributes,
1726      att.global.analytic.attributes,
1727      att.global.facs.attributes,
1728      att.pointing.attributes,
1729      empty
1730    }
```

```
1731 revisionDesc =
1732
1733   ## (revision description) summarizes the revision history for a
1734   ## file.
1735   element tei:revisionDesc {
1736     (\list | change+),
1737     att.global.attribute.xmlid,
1738     att.global.attribute.xmlspace,
1739     att.global.linking.attributes,
1740     att.global.analytic.attributes,
1741     att.global.facs.attributes,
1742     att.docStatus.attributes,
1743     empty
1744   }
1745 change =
1746
1747   ## summarizes a particular change or correction made to a particular
1748   ## version of an electronic text which is shared between several
1749   ## researchers.
1750   element tei:change {
1751     (text | model.limitedPhrase | model.inter | model.global)*,
1752     att.global.attribute.xmlid,
1753     att.global.attribute.n,
1754     att.global.attribute.xmlspace,
1755     att.global.linking.attributes,
1756     att.global.analytic.attributes,
1757     att.global.facs.attributes,
1758     att.ascribed.attributes,
1759     att.datable.w3c.attributes,
1760     att.datable.iso.attributes,
1761     att.docStatus.attributes,
1762     empty
1763   }
1764 anchor =
1765
1766   ## (anchor point) attaches an identifier to a point within a text,
1767   ## whether or not it corresponds with a textual element.
1768   element tei:anchor {
1769     empty,
1770     att.global.attribute.xmlid,
1771     att.global.attribute.xmlspace,
1772     att.global.linking.attributes,
1773     att.global.analytic.attributes,
1774     att.global.facs.attributes,
1775     att.typed.attribute.type,
1776     empty
1777   }
1778 seg =
1779
1780   ## (arbitrary segment) represents any segmentation of text below the
1781   ## chunk level.
1782   element tei:seg {
1783     macro.paraContent,
1784     att.global.attribute.xmlid,
1785     att.global.attribute.n,
1786     att.global.attribute.xmlspace,
1787     att.global.linking.attributes,
1788     att.global.analytic.attributes,
1789     att.global.facs.attributes,
1790     att.typed.attribute.type,
1791     att.responsibility.attributes,
1792     empty
1793   }
```

```
1794 att.global.linking.attributes =
1795   att.global.linking.attribute.corresp ,
1796   att.global.linking.attribute.synch ,
1797   att.global.linking.attribute.sameAs
1798 att.global.linking.attribute.corresp =
1799
1800   ## (corresponds) points to elements that correspond to the current
1801   ## element in some way.
1802   attribute corresp {
1803     list { xsd:anyURI, xsd:anyURI* }
1804   }?
1805 att.global.linking.attribute.synch =
1806
1807   ## (synchronous) points to elements that are synchronous with the
1808   ## current element.
1809   attribute synch {
1810     list { xsd:anyURI, xsd:anyURI* }
1811   }?
1812 att.global.linking.attribute.sameAs =
1813
1814   ## points to an element that is the same as the current element.
1815   attribute sameAs { xsd:anyURI }?
1816 orgName =
1817
1818   ## (organization name) contains an organizational name.
1819   element tei:orgName {
1820     macro.phraseSeq ,
1821     att.global.attribute.xmlid ,
1822     att.global.attribute.xmlspace ,
1823     att.global.linking.attributes ,
1824     att.global.analytic.attributes ,
1825     att.global.facs.attributes ,
1826     att.datable.w3c.attribute.when ,
1827     att.datable.iso.attributes ,
1828     att.editLike.attributes ,
1829     att.personal.attributes ,
1830     empty
1831   }
1832 persName =
1833
1834   ## (personal name) contains a proper noun or proper-noun phrase
1835   ## referring to a person, possibly including any or all of the
1836   ## person's forenames, surnames, honorifics, added names, etc.
1837   element tei:persName {
1838     macro.phraseSeq ,
1839     att.global.attribute.xmlid ,
1840     att.global.attribute.xmlspace ,
1841     att.global.linking.attributes ,
1842     att.global.analytic.attributes ,
1843     att.global.facs.attributes ,
1844     att.datable.w3c.attribute.when ,
1845     att.datable.iso.attributes ,
1846     att.editLike.attributes ,
1847     att.personal.attributes ,
1848     empty
1849   }
1850 age =
1851
1852   ## (age) specifies the age of a person.
1853   element tei:age {
1854     macro.phraseSeq.limited ,
1855
1856     ## supplies a numeric code representing the age or age group
```

```
1857      attribute value { xsd:nonNegativeInteger }?,
1858      att.global.attribute.xmlid ,
1859      att.global.attribute.xmlspace ,
1860      att.global.linking.attributes ,
1861      att.global.analytic.attributes ,
1862      att.global.facs.attributes ,
1863      att.editLike.attributes ,
1864      att.datable.w3c.attribute.when ,
1865      att.datable.iso.attributes ,
1866      empty
1867    }
1868  langKnowledge =
1869
1870    ## (language knowledge) summarizes the state of a person's
1871    ## linguistic knowledge, either as prose or by a list of langKnown
1872    ## elements.
1873    element tei:langKnowledge {
1874      (model.pLike | langKnown+),
1875      att.global.attribute.xmlid ,
1876      att.global.attribute.xmlspace ,
1877      att.global.linking.attributes ,
1878      att.global.analytic.attributes ,
1879      att.global.facs.attributes ,
1880      att.datable.w3c.attribute.when ,
1881      att.datable.iso.attributes ,
1882      att.editLike.attributes ,
1883      empty
1884    }
1885  langKnown =
1886
1887    ## (language known) summarizes the state of a person's linguistic
1888    ## competence, i.e., knowledge of a single language.
1889    element tei:langKnown {
1890      macro.phraseSeq.limited ,
1891
1892      ## supplies a valid language tag for the language concerned.
1893      attribute tag { xsd:language },
1894
1895      ## a code indicating the person's level of knowledge for this
1896      ## language
1897      attribute level {
1898        xsd:token { pattern = "(\p{L}|\p{N}|\p{P}|\p{S})+" }
1899      }?,
1900      att.global.attribute.xmlid ,
1901      att.global.attribute.xmlspace ,
1902      att.global.linking.attributes ,
1903      att.global.analytic.attributes ,
1904      att.global.facs.attributes ,
1905      att.datable.w3c.attribute.when ,
1906      att.datable.iso.attributes ,
1907      att.editLike.attributes ,
1908      empty
1909    }
1910  listPerson =
1911
1912    ## (list of persons) contains a list of descriptions, each of which
1913    ## provides information about an identifiable person or a group of
1914    ## people, for example the participants in a language interaction,
1915    ## or the people referred to in a historical source.
1916    element tei:listPerson {
1917      (model.headLike*,
1918       (model.personLike | listPerson)+,
1919       (relation | relationGrp)*),
```

```
1920      att.global.attribute.xmlid ,
1921      att.global.attribute.xmlspace ,
1922      att.global.linking.attributes ,
1923      att.global.analytic.attributes ,
1924      att.global.facs.attributes ,
1925      att.typed.attribute.type ,
1926      att.declarable.attributes ,
1927      empty
1928    }
1929  occupation =
1930
1931    ## contains an informal description of a person's trade, profession
1932    ## or occupation.
1933    element tei:occupation {
1934      macro.phraseSeq ,
1935      att.global.attribute.xmlid ,
1936      att.global.attribute.xmlspace ,
1937      att.global.linking.attributes ,
1938      att.global.analytic.attributes ,
1939      att.global.facs.attributes ,
1940      att.datable.w3c.attribute.when ,
1941      att.datable.iso.attributes ,
1942      att.editLike.attributes ,
1943      att.naming.attributes ,
1944      empty
1945    }
1946  relationGrp =
1947
1948    ## (relation group) provides information about relationships
1949    ## identified amongst people, places, and organizations, either
1950    ## informally as prose or as formally expressed relation links.
1951    element tei:relationGrp {
1952      (model.pLike+ | relation+),
1953      att.global.attribute.xmlid ,
1954      att.global.attribute.xmlspace ,
1955      att.global.linking.attributes ,
1956      att.global.analytic.attributes ,
1957      att.global.facs.attributes ,
1958      empty
1959    }
1960  person =
1961
1962    ## provides information about an identifiable individual, for
1963    ## example a participant in a language interaction, or a person
1964    ## referred to in a historical source.
1965    element tei:person {
1966      (model.pLike+ | (model.personPart | model.global)*),
1967
1968      ## specifies a primary role or classification for the person.
1969      attribute role {
1970        list {
1971          (
1972            "audience"
1973            |
1974              "chair"
1975            |
1976              "coordinator"
1977            |
1978              "interviewee"
1979            |
1980              "interviewer"
1981            |
1982              "non-participant"
```

```
1983            |
1984              "participant"
1985            |
1986              "presenter"
1987            |
1988              "researcher"
1989            |
1990              "student"
1991            |
1992              "teacher"
1993            |
1994              "translator"),
1995           (
1996            "audience"
1997            |
1998              "chair"
1999            |
2000              "coordinator"
2001            |
2002              "interviewee"
2003            |
2004              "interviewer"
2005            |
2006              "non-participant"
2007            |
2008              "participant"
2009            |
2010              "presenter"
2011            |
2012              "researcher"
2013            |
2014              "student"
2015            |
2016              "teacher"
2017            |
2018              "translator")*
2019         }
2020       }?,
2021
2022       ## specifies the sex of the person.
2023       attribute sex { "0" | "1" | "2" | "9" }?,
2024       att.global.attribute.xmlid ,
2025       att.global.attribute.xmlspace ,
2026       att.global.linking.attributes ,
2027       att.global.analytic.attributes ,
2028       att.global.facs.attributes ,
2029       att.editLike.attributes ,
2030       empty
2031     }
2032 personGrp =
2033
2034   ## (personal group) describes a group of individuals treated as a
2035   ## single person for analytic purposes.
2036   element tei:personGrp {
2037     (model.pLike+ | model.personPart*),
2038
2039     ## specifies the role of this group of participants in the
2040     ## interaction.
2041     attribute role {
2042
2043       "audience"
2044       |
2045         "interactants"
```

```
      |
        "speakers"
    }?,

    ## specifies the size or approximate size of the group.
    attribute size {
      list {
        xsd:token { pattern = "(\p{L}|\p{N}|\p{P}|\p{S})+" },
        xsd:token { pattern = "(\p{L}|\p{N}|\p{P}|\p{S})+" }*
      }
    }?,
    att.global.attribute.xmlid,
    att.global.attribute.xmlspace,
    att.global.linking.attributes,
    att.global.analytic.attributes,
    att.global.facs.attributes,
    empty
  }
relation =

  ## (relationship) describes any kind of relationship or linkage
  ## amongst a specified group of participants.
  element tei:relation {
    empty
    >> sch:pattern [
         id = "relation-constraint-activemutual"
         "\x{a}" ~
         "          "
         sch:rule [
           context = "tei:relation"
           "\x{a}" ~
           "                "
           sch:report [
             test = "@active and @mutual"
             "Only one of the attributes\x{a}" ~
             "  'active' and 'mutual' may be supplied"
           ]
           "\x{a}" ~
           "            "
         ]
         "\x{a}" ~
         "         "
       ]
    >> sch:pattern [
         id = "relation-constraint-activepassive"
         "\x{a}" ~
         "           "
         sch:rule [
           context = "tei:relation"
           "\x{a}" ~
           "                "
           sch:report [
             test = "@passive and not(@active)"
             "the attribute 'passive'\x{a}" ~
             "  may be supplied only if the attribute 'active' is\x{a}" ~
             "  supplied"
           ]
           "\x{a}" ~
           "            "
         ]
         "\x{a}" ~
         "         "
       ],
```

```
2109      (
2110       ## supplies a list of participants amongst all of whom the
2111       ## relationship holds equally.
2112       attribute mutual {
2113         list { xsd:anyURI, xsd:anyURI* }
2114       }?),
2115
2116      ## categorizes the relationship in some respect, e.g. as social,
2117      ## personal or other.
2118      [ a:defaultValue = "personal" ]
2119      attribute type {
2120
2121        "acquaintedness"
2122        |
2123          "power"
2124      }?,
2125
2126      ## supplies a name for the kind of relationship of which this is
2127      ## an instance.
2128      attribute name {
2129
2130        "acquainted"
2131        |
2132          "acquaintedness_unknown"
2133        |
2134          "power_relations_unknown"
2135        |
2136          "predominantly_acquainted"
2137        |
2138          "predominantly_unacquainted"
2139        |
2140          "fairly_asymmetrical"
2141        |
2142          "fairly_symmetrical"
2143        |
2144          "unacquainted"
2145      },
2146      att.global.attribute.xmlid ,
2147      att.global.attribute.xmlspace ,
2148      att.global.linking.attributes ,
2149      att.global.analytic.attributes ,
2150      att.global.facs.attributes ,
2151      att.datable.w3c.attribute.when ,
2152      att.datable.iso.attributes ,
2153      att.editLike.attributes ,
2154      att.naming.attributes ,
2155      empty
2156    }
2157  sex =
2158
2159    ## specifies the sex of a person.
2160    element tei:sex {
2161      macro.phraseSeq ,
2162
2163      attribute value { "0" | "1" | "2" | "9" }?,
2164      att.global.attribute.xmlid ,
2165      att.global.attribute.xmlspace ,
2166      att.global.linking.attributes ,
2167      att.global.analytic.attributes ,
2168      att.global.facs.attributes ,
2169      att.editLike.attributes ,
2170      att.datable.w3c.attribute.when ,
2171      att.datable.iso.attributes ,
```

```
2172      empty
2173    }
2174  att.datable.iso.attributes = empty
2175  att.duration.iso.attributes = empty
2176  model.global.spoken = pause | vocal | incident | shift
2177  model.global.spoken_alternation = pause | vocal | incident | shift
2178  model.global.spoken_sequence = pause, vocal, incident, shift
2179  model.global.spoken_sequenceOptional = pause?, vocal?, incident?, shift?
2180  model.global.spoken_sequenceOptionalRepeatable =
2181    pause*, vocal*, incident*, shift*
2182  model.global.spoken_sequenceRepeatable =
2183    pause+, vocal+, incident+, shift+
2184  model.recordingPart = model.dateLike | respStmt | equipment
2185  scriptStmt =
2186
2187    ## (script statement) contains a citation giving details of the
2188    ## script used for a spoken text.
2189    element tei:scriptStmt {
2190      (model.pLike+ | model.biblLike),
2191      att.global.attribute.xmlid,
2192      att.global.attribute.xmlspace,
2193      att.global.linking.attributes,
2194      att.global.analytic.attributes,
2195      att.global.facs.attributes,
2196      att.declarable.attributes,
2197      empty
2198    }
2199  recordingStmt =
2200
2201    ## (recording statement) describes a set of recordings used as the
2202    ## basis for transcription of a spoken text.
2203    element tei:recordingStmt {
2204      (model.pLike+ | recording+),
2205      att.global.attribute.xmlid,
2206      att.global.attribute.xmlspace,
2207      att.global.linking.attributes,
2208      att.global.analytic.attributes,
2209      att.global.facs.attributes,
2210      empty
2211    }
2212  recording =
2213
2214    ## (recording event) details of an audio or video recording event
2215    ## used as the source of a spoken text, either directly or from a
2216    ## public broadcast.
2217    element tei:recording {
2218      (model.pLike+ | model.recordingPart*),
2219
2220      ## the kind of recording.
2221      [ a:defaultValue = "audio" ]
2222      attribute type {
2223
2224        "audio"
2225      }?,
2226      att.global.attribute.xmlid,
2227      att.global.attribute.xmlspace,
2228      att.global.linking.attributes,
2229      att.global.analytic.attributes,
2230      att.global.facs.attributes,
2231      att.declarable.attributes,
2232      att.duration.w3c.attributes,
2233      att.duration.iso.attributes,
2234      empty
```

```
2235    }
2236  equipment =
2237
2238    ## provides technical details of the equipment and media used for an
2239    ## audio or video recording used as the source for a spoken text.
2240    element tei:equipment {
2241      model.pLike+,
2242      att.global.attribute.xmlid,
2243      att.global.attribute.xmlspace,
2244      att.global.linking.attributes,
2245      att.global.analytic.attributes,
2246      att.global.facs.attributes,
2247      att.declarable.attributes,
2248      empty
2249    }
2250  u =
2251
2252    ## (utterance) a stretch of speech usually preceded and followed by
2253    ## silence or by a change of speaker.
2254    element tei:u {
2255      (text | model.gLike | model.phrase | model.global)*,
2256      att.global.attribute.xmlid,
2257      att.global.attribute.xmlspace,
2258      att.global.linking.attributes,
2259      att.global.analytic.attributes,
2260      att.global.facs.attributes,
2261      att.declaring.attributes,
2262      att.ascribed.attributes,
2263      empty
2264    }
2265  pause =
2266
2267    ## a pause either between or within utterances.
2268    element tei:pause {
2269      empty,
2270      att.global.attribute.xmlid,
2271      att.global.attribute.xmlspace,
2272      att.global.linking.attributes,
2273      att.global.analytic.attributes,
2274      att.global.facs.attributes,
2275      att.timed.attributes,
2276      empty
2277    }
2278  vocal =
2279
2280    ## any vocalized but not necessarily lexical phenomenon, for example
2281    ## voiced pauses, non-lexical backchannels, etc.
2282    element tei:vocal {
2283      model.glossLike*,
2284
2285      ## Specifies the approx. count of syllables
2286      attribute voice:syl { xsd:nonNegativeInteger }?,
2287
2288      attribute voice:desc { text }?,
2289
2290      ## indicates whether or not the phenomenon is repeated.
2291      [ a:defaultValue = "false" ]
2292      attribute iterated { xsd:boolean | "unknown" | "inapplicable" }?,
2293      att.global.attribute.xmlid,
2294      att.global.attribute.xmlspace,
2295      att.global.linking.attributes,
2296      att.global.analytic.attributes,
2297      att.global.facs.attributes,
```

```
2298      att.timed.attributes,
2299      att.typed.attributes,
2300      empty
2301    }
2302  incident =
2303
2304    ## any phenomenon or occurrence, not necessarily vocalized or
2305    ## communicative, for example incidental noises or other events
2306    ## affecting communication.
2307    element tei:incident {
2308      model.glossLike*,
2309
2310      attribute voice:desc { text }?,
2311      att.global.attribute.xmlid,
2312      att.global.attribute.xmlspace,
2313      att.global.linking.attributes,
2314      att.global.analytic.attributes,
2315      att.global.facs.attributes,
2316      att.timed.attributes,
2317      empty
2318    }
2319  shift =
2320
2321    ## marks the point at which some paralinguistic feature of a series
2322    ## of utterances by any one speaker changes.
2323    element tei:shift {
2324      empty,
2325
2326      ## a paralinguistic feature.
2327      attribute feature {
2328
2329        "custom"
2330        |
2331          "loud"
2332        |
2333          "tempo"
2334        |
2335          "voice"
2336      }?,
2337
2338      ## specifies the new state of the paralinguistic feature
2339      ## specified.
2340      [ a:defaultValue = "normal" ]
2341      attribute new {
2342
2343        "a"
2344        |
2345          "f"
2346        |
2347          "high"
2348        |
2349          "imitating"
2350        |
2351          "l"
2352        |
2353          "laugh"
2354        |
2355          "neutral"
2356        |
2357          "not"
2358        |
2359          "p"
2360        |
```

```
2361            "phone"
2362          |
2363            "reading"
2364          |
2365            "reading_aloud"
2366          |
2367            "sigh"
2368          |
2369            "singing"
2370          |
2371            "speaking"
2372          |
2373            "voice"
2374          |
2375            "whisp"
2376          |
2377            "with"
2378          |
2379            "yawning"
2380          |
2381            "speaking_with_a_high_voice"
2382        }?,
2383        att.global.attribute.xmlid ,
2384        att.global.attribute.xmlspace ,
2385        att.global.linking.attributes ,
2386        att.global.analytic.attributes ,
2387        att.global.facs.attributes ,
2388        empty
2389      }
2390  model.divPart.spoken = u
2391  model.divPart.spoken_alternation = u
2392  model.divPart.spoken_sequence = u
2393  model.divPart.spoken_sequenceOptional = u?
2394  model.divPart.spoken_sequenceOptionalRepeatable = u*
2395  model.divPart.spoken_sequenceRepeatable = u+
2396  att =
2397
2398    ## (attribute) contains the name of an attribute appearing within
2399    ## running text.
2400    element tei:att {
2401      text ,
2402      att.global.attribute.xmlid ,
2403      att.global.attribute.xmlspace ,
2404      att.global.linking.attributes ,
2405      att.global.analytic.attributes ,
2406      att.global.facs.attributes ,
2407      empty
2408    }
2409  gi =
2410
2411    ## (element name) contains the name (generic identifier) of an
2412    ## element.
2413    element tei:gi {
2414      text ,
2415      att.global.attribute.xmlid ,
2416      att.global.attribute.xmlspace ,
2417      att.global.linking.attributes ,
2418      att.global.analytic.attributes ,
2419      att.global.facs.attributes ,
2420      empty
2421    }
2422  val =
2423
```

```
2424    ## (value) contains a single attribute value.
2425    element tei:val {
2426      text,
2427      att.global.attribute.xmlid,
2428      att.global.attribute.xmlspace,
2429      att.global.linking.attributes,
2430      att.global.analytic.attributes,
2431      att.global.facs.attributes,
2432      empty
2433    }
2434  classRef =
2435
2436    ## points to the specification for an attribute or model class which
2437    ## is to be included in a schema
2438    element tei:classRef {
2439      empty,
2440      att.global.attributes,
2441      att.readFrom.attributes,
2442
2443      ## the identifier used for the required class within the source
2444      ## indicated.
2445      attribute key { xsd:NCName },
2446      empty
2447    }
2448  elementRef =
2449
2450    ## points to the specification for some element which is to be
2451    ## included in a schema
2452    element tei:elementRef {
2453      empty,
2454      att.global.attributes,
2455      att.readFrom.attributes,
2456
2457      ## the identifier used for the required element within the source
2458      ## indicated.
2459      attribute key { xsd:NCName },
2460      empty
2461    }
2462  macroRef =
2463
2464    ## points to the specification for some pattern which is to be
2465    ## included in a schema
2466    element tei:macroRef {
2467      empty,
2468      att.global.attributes,
2469      att.readFrom.attributes,
2470
2471      ## the identifier used for the required pattern within the source
2472      ## indicated.
2473      attribute key { xsd:NCName },
2474      empty
2475    }
2476  constraint =
2477
2478    ## (constraint rules) the formal rules of a constraint
2479    element tei:constraint {
2480      (text | macro.anyXML), att.global.attributes, empty
2481    }
2482  att.combinable.attributes = att.combinable.attribute.mode
2483  att.combinable.attribute.mode =
2484
2485    ## specifies the effect of this declaration on its parent object.
2486    [ a:defaultValue = "add" ]
```

```
2487    attribute mode {
2488
2489      ## this declaration is added to the current definitions
2490      "add"
2491      |
2492        ## if present already, the whole of the declaration for this
2493        ## object is removed from the current setup
2494        "delete"
2495      |
2496        ## this declaration changes the declaration of the same name in
2497        ## the current definition
2498        "change"
2499      |
2500        ## this declaration replaces the declaration of the same name in
2501        ## the current definition
2502        "replace"
2503    }?
2504  att.identified.attributes = att.combinable.attributes
2505  TEI =
2506
2507    ## (TEI document) contains a single TEI-conformant document,
2508    ## comprising a TEI header and a text, either in isolation or as
2509    ## part of a teiCorpus element.
2510    element tei:TEI {
2511      (teiHeader,
2512       ((model.resourceLike+, \text?) | \text)),
2513
2514      ## specifies the version number of the TEI Guidelines against
2515      ## which this document is valid.
2516      attribute version {
2517        xsd:token { pattern = "[\d]+(\.[\d]+){0,2}" }
2518      }?,
2519      att.global.attribute.xmlid,
2520      att.global.attribute.xmllang,
2521      att.global.attribute.xmlbase,
2522      att.global.attribute.xmlspace,
2523      att.global.linking.attributes,
2524      att.global.analytic.attributes,
2525      att.global.facs.attributes,
2526      empty
2527    }
2528  \text =
2529
2530    ## contains a single text of any kind, whether unitary or composite,
2531    ## for example a poem or drama, a collection of essays, a novel, a
2532    ## dictionary, or a corpus sample.
2533    element tei:text {
2534      (model.global*, body, model.global*),
2535      att.global.attribute.xmlid,
2536      att.global.attribute.xmlspace,
2537      att.global.linking.attributes,
2538      att.global.analytic.attributes,
2539      att.global.facs.attributes,
2540      att.declaring.attributes,
2541      empty
2542    }
2543  body =
2544
2545    ## (text body) contains the whole body of a single unitary text,
2546    ## excluding any front or back matter.
2547    element tei:body {
2548      (model.global*,
2549       (model.divTop, (model.global | model.divTop)*)?,
```

```
2550        (model.divGenLike, (model.global | model.divGenLike)*)?,
2551        ((model.divLike, (model.global | model.divGenLike)*)+
2552         | (model.div1Like, (model.global | model.divGenLike)*)+
2553         | ((model.common, model.global*)+,
2554            ((model.divLike, (model.global | model.divGenLike)*)+
2555             | (model.div1Like, (model.global | model.divGenLike)*)+)?)),
2556        (model.divBottom, model.global*)*),
2557      att.global.attribute.xmlid,
2558      att.global.attribute.xmlspace,
2559      att.global.linking.attributes,
2560      att.global.analytic.attributes,
2561      att.global.facs.attributes,
2562      att.declaring.attributes,
2563      empty
2564    }
2565  \div =
2566
2567    ## (text division) contains a subdivision of the front, body, or
2568    ## back of a text.
2569    element tei:div {
2570      ((model.divTop | model.global)*,
2571       ((((model.divLike | model.divGenLike), model.global*)+
2572         | ((model.common, model.global*)+,
2573            ((model.divLike | model.divGenLike), model.global*)*)),
2574        (model.divBottom, model.global*)*)?),
2575
2576      attribute voice:end { text }?,
2577
2578      attribute voice:start { text }?,
2579
2580      attribute voice:medium_type { text }?,
2581
2582      attribute voice:end_medium_number { text }?,
2583
2584      attribute voice:start_medium_number { text }?,
2585
2586      attribute voice:end_track { text }?,
2587
2588      attribute voice:start_track { text }?,
2589      att.global.attribute.xmlid,
2590      att.global.attribute.xmlspace,
2591      att.global.linking.attributes,
2592      att.global.analytic.attributes,
2593      att.global.facs.attributes,
2594      att.divLike.attributes,
2595      att.typed.attribute.type,
2596      att.declaring.attributes,
2597      empty
2598    }
2599  att.global.facs.attributes = empty
2600  supplied =
2601
2602    ## signifies text supplied by the transcriber or editor for any
2603    ## reason, typically because the original cannot be read because of
2604    ## physical damage or loss to the original.
2605    element tei:supplied {
2606      macro.paraContent,
2607
2608      ## indicates why the text has had to be supplied.
2609      attribute reason {
2610        list {
2611          xsd:token { pattern = "(\p{L}|\p{N}|\p{P}|\p{S})+" },
2612          xsd:token { pattern = "(\p{L}|\p{N}|\p{P}|\p{S})+" }*
```

```
2613        }
2614      }?,
2615      att.global.attribute.xmlid ,
2616      att.global.attribute.xmlspace ,
2617      att.global.linking.attributes ,
2618      att.global.analytic.attributes ,
2619      att.global.facs.attributes ,
2620      att.editLike.attributes ,
2621      empty
2622    }
2623  surplus =
2624
2625    ## (Texte superflu) marks text present in the source which the
2626    ## editor believes to be superfluous or redundant.
2627    element tei:surplus {
2628      macro.paraContent ,
2629      att.global.attributes ,
2630      att.editLike.attributes ,
2631
2632      ## indicates the grounds for believing this text to be
2633      ## superfluous.
2634      attribute reason {
2635        list {
2636          xsd:token { pattern = "(\p{L}|\p{N}|\p{P}|\p{S})+" },
2637          xsd:token { pattern = "(\p{L}|\p{N}|\p{P}|\p{S})+" }*
2638        }
2639      }?,
2640      empty
2641    }
2642  track =
2643
2644    ## indicating the beginning of a new track.
2645    element voice:track {
2646      empty ,
2647      att.global.attributes ,
2648      att.typed.attributes ,
2649
2650      attribute medium { xsd:string }?,
2651
2652      attribute num { xsd:string }?,
2653      empty
2654    }
2655  pvc =
2656
2657    ## Indicating a Pronunciation Variation and Coinage as described in
2658    ## the VOICE Transcription Conventions
2659    element voice:pvc {
2660      macro.paraContent ,
2661      att.global.attributes ,
2662
2663      attribute comment { text }?,
2664      empty
2665    }
2666  to =
2667
2668    ## Indicating a stretch of speach directed to a person
2669    element voice:to {
2670      macro.paraContent ,
2671      att.segLike.attributes ,
2672      att.ascribed.attributes ,
2673      empty
2674    }
2675  start = TEI | teiCorpus
```

# D VOICE Transcription Conventions

Much of this thesis draws from and was developed alongside with the VOICE Transcription Conventions. For this reason, this part of the appendix provides this set of documents. The summary below is taken verbatim from the official public documentation of the VOICE transcription system (cf. http://www.univie.ac.at/voice/page/transcription_general_information).
The following sections duplicate the documents[44] that constitute the VOICE Transcription Conventions and are included to provide a convenient point of reference (see *Mark-up Conventions (p.219)* and *Spelling Conventions p.229*).

Transcription conventions in a project like VOICE are of particular importance and need to reconcile three main requirements: 1) they need to capture the reality of spoken interactions as precisely as possible, 2) they need to be replicable, i.e. the scheme must be usable without further explanation by other researchers, 3) they need to make sure that the resulting transcriptions are computer-readable. As with any other transcription conventions, the reconciliation of these different requirements calls for compromise and some aspects of spoken interaction, for example, many of its non-vocal paralinguistic features, necessarily fall outside its scope.

The VOICE transcription conventions, which are the result of our extensive experience in applying these criteria to a wide range of ELF data, are of two kinds: mark-up and spelling. The VOICE mark-up conventions are specifically designed to reflect what seem to be the most significant features of ELF interactions. For instance, the nature of our data prompted us to devise a fairly detailed set of descriptors for pronunciation variations and coinages, for code-switching, for onomatopoeic sounds and for laughter, not only as such but as a prosodic feature of speech. The VOICE spelling conventions are designed to render the diversity of ELF speech in a standardized way.

These transcription conventions are, of course, subject to revision as our research proceeds. They are made available here with a view to facilitating the understanding of VOICE transcripts. However, other ELF researchers are invited to make use of the conventions for their own research.

## D.1 Mark-up Conventions



Version 2.1 June 2007

[44] The documents are re-formatted to fit the style of this thesis more closely

### D.1.1 SPEAKER IDS

| Example | Description |
|---|---|
| S1:<br>S2:<br>... | Speakers are generally numbered in the order they first speak. The speaker ID is given at the beginning of each turn. |
| SS: | Utterances assigned to more than one speaker (e.g. an audience), spoken either in unison or staggered, are marked with a collective speaker ID **SS**. |
| SX: | Utterances that cannot be assigned to a particular speaker are marked **SX**. |
| SX-f:<br>SX-m: | Utterances that cannot be assigned to a particular speaker, but where the gender can be identified, are marked **SX-f** or **SX-m**. |
| SX-1:<br>SX-2:<br>... | If it is likely but not certain that a particular speaker produced the utterance in question, this is marked **SX-1**, **SX-2**, etc. |

### D.1.2 INTONATION

| Example | Description |
|---|---|
| S1: that's what my next er slide**?** does | Words spoken with rising intonation are followed by a question mark "**?**" . |
| S7: that's point two**.** absolutely yes**.** | Words spoken with falling intonation are followed by a full stop "**.**" . |

### D.1.3 EMPHASIS

| Example | Description |
|---|---|
| S7: er internationalization is a very IMPORTANT issue<br>S3: toMORrow we have to work on the presentation already | If a speaker gives a syllable, word or phrase particular prominence, this is written in capital letters. |

### D.1.4 PAUSES

| Example | Description |
|---|---|
| SX-f: because they all give me different **(.)** different **(.)** points of view | Every brief pause in speech (up to a good half second) is marked with a full stop in parentheses. |
| S1: aha **(2)** so finally arrival on monday evening is still valid | Longer pauses are timed to the nearest second and marked with the number of seconds in parentheses, e.g. (1) = 1 second, (3) = 3 seconds. |

### D.1.5 OVERLAPS

| Example | Description |
|---|---|
| S1: it is your best **<1> case </1>** scenario (.)<br>S2: **<1> yeah </1>**<br>S1: okay | Whenever two or more utterances happen at the same time, the overlaps are marked with numbered tags: **<1> </1>**, **<2> </2>**,... Everything that is simultaneous gets the same number. All overlaps are marked in blue. |
| S9: it it is (.) to identify **some<1>thing** </1> where (.)<br>S3: <1> mhm </1> | All overlaps are approximate and words may be split up if appropriate. In this case, the tag is placed within the split-up word. |

### D.1.6 OTHER-CONTINUATION

| Example | Description |
|---|---|
| S1: what up till (.) till twelve?<br>S2: **yes=**<br>S1: **=really**. so it's it's quite a lot of time. | Whenever a speaker continues, completes or supports another speaker's turn immediately (i.e. without a pause), this is marked by "". |

### D.1.7 LENGTHENING

| Example | Description |
|---|---|
| S1: you can run faster but they have much **mo:re** technique with the ball<br>S5: personally that's my opinion the: **er::m** | Lengthened sounds are marked with a colon "**:**".<br>Exceptionally long sounds (i.e. approximating 2 seconds or more) are marked with a double colon "**::**". |

### D.1.8 REPETITION

| Example | Description |
|---|---|
| S11: e:r i'd like to go t- t- **to to** this type of course | All repetitions of words and phrases (including self-interruptions and false starts) are transcribed. |

### D.1.9 WORD FRAGMENTS

| Example | Description |
|---|---|
| S6: with a minimum of (.) of **participa-**<br>S1: mhm<br>S6: **-pation** from french universities to say we have er (.) a joint doctorate or a **joi-** joint master | With word fragments, a hyphen marks where a part of the word is missing. |

### D.1.10 LAUGHTER

| Example | Description |
|---|---|
| S1: in denmark well who knows. @@<br>S2: **<@>** yeah **</@>** @@ that's right | All laughter and laughter-like soundsare transcribed with the @ symbol, approximating syllable number (e.g. ha ha ha = @@@). Utterances spoken laughingly are put between **<@> </@>** tags. |

### D.1.11 UNCERTAIN TRANSCRIPTION

| Example | Description |
|---|---|
| S3: i've a lot of very **(generous)** friends<br>SX-4: they will do whatever they want because they are a compan**(ies)** | Word fragments, words or phrases which cannot be reliably identified are put in parentheses ( ). |

### D.1.12 PRONUNCIATION VARIATIONS & COINAGES

| Example | Description |
|---|---|
| S4: i also: (.) e:r played (.) tennis e:r **<pvc> bices </pvc>** e:r we rent? went? | Striking variations on the levels of phonology, morphology and lexis as well as 'invented' words are marked **<pvc> </pvc>**. |
| S9: how you were controlling such a thing and how you **<pvc> (avrivate) </pvc>** (it) | What you hear is represented in spelling according to general principles of English orthography. Uncertain transcription is put in parentheses **( )** . |
| S6: what we try to explain here is the foreign direct investment growth (2) in a certain industry (.) and a certain **<pvc>** compy **{company}</pvc>** | If a corresponding existing word can be identified, this existing word isadded between curly brackets **{ }**. |
| S2: anyway i make you an a total (.) **<pvc>**summamary {summary} **<ipa>** sʌməˈmærɪ**</ipa> </pvc>** of destinations | Particularly when it comes to salient variations on the level of phonology, e.g. sound substitution or addition, a phonetic representation should be added between **<ipa> </ipa>** tags. |

### D.1.13 ONOMATOPOEIC NOISES

| Example | Description |
|---|---|
| S1: it may be quite HARMLESS and at the end of the day you (.) **<ono>**dəʃdəʃdəʃ**</ono>** (.) somebody | When speakers produce noises in order to imitate something instead of using words, these onomatopoeic noises are rendered in IPA symbols between **<ono> </ono>** tags. |

### D.1.14 NON-ENGLISH SPEECH

| Example | Description |
|---|---|
| S5: **<L1de> bei firmen </L1de>** or wherever | Utterances in a participant's first language (L1) are put between tags indicating the speaker's **L1**. |
| S7: er this is **<LNde> die seite? (welche) </LNde>** is | Utterances in languages which are neither English nor the speaker's first language are marked **LN** with the language indicated. |
| S4: it depends in in in **<LQit> roma </LQit>** | Non-English utterances where it cannot be ascertained whether the language is the speaker's first language or a foreign language are marked **LQ** with the language indicated. |

| | |
|---|---|
| S2: erm we want to go t- to **<LNvi> xx xxx </LNvi>** island first of all | Unintelligible utterances in a participant's L1, LN or in an LQ are represented by **x's** approximating syllable number. |
| S4: and now we do the boat trip (1) **<L1xx> xxxxx </L1xx>**<br>S3: mhm | Utterances in a language onecannot recognize are marked **L1xx**, **LNxx** or **LQxx**. |
| S3: <L1fr> oui un grand carre **{yes like a big square}** </L1fr> (.) i <fast> think it would </fast> be better if we put the tables a <soft> different way </soft> | If possible, translations into English are provided between curly brackets **{ }** immediately after the non-English speech. |

### D.1.15 SPELLING OUT

| Example | Description |
|---|---|
| S1: and they (3) created some (1) some er (2) JARGON. do you know? the word JARGON? (.) **<spel> j a r- </spel> <spel> j a r g o n? </spel>** jargon | The **<spel> </spel>** tag is used to mark words or abbreviations which are spelled out by the speaker, i.e. words whose constituents are pronounced as individual letters. |

### D.1.16 SPEAKING MODES

| Example | Description |
|---|---|
| S2: because as i explained before is that we have in the **<fast>** universities of cyprus we have **</fast>** a specific e:rm procedure<br><fast> </fast><br><slow> </slow><br><loud> </loud><br><soft> </soft><br><whispering> </whispering><br><sighing> </sighing><br><reading> </reading><br><reading aloud> </reading aloud><br><on phone> </on phone><br><imitating> </imitating><br><singing> </singing><br><yawning> </yawning> | Utterances which are spoken in a particular mode (fast, soft, whispered, read, etc.) and are notably different from the speaker's normal speaking style are marked accordingly.<br>The list of speaking modes is an open one. |

### D.1.17 BREATH

| Example | Description |
|---|---|
| S1: so it's always **hh** (.) going around (2) yeah | Noticeable breathing in or out is represented by two or three **h's** (hh = relatively short; hhh= relatively long). |

### D.1.18 SPEAKER NOISES

| Example | Description |
|---|---|
| <coughs><br><clears throat><br><sniffs><br><sneezes><br><snorts><br><applauds><br><smacks lips><br><yawns><br><whistles><br><swallows> | Noises produced by the current speaker are always transcribed. Noises produced by other speakers are only transcribed if they seem relevant (e.g. because they make speech unintelligible or influence the interaction).<br>The list of speaker noises is an open one. |
| S1: yeah <1> what </1> i think in in doctor levels<br>S7: **<clears throat>** </1> | These noises are transcribed as part of the running text and put between pointed brackets **< >.** |
| SX-m: but you NEVER KNOW when it's popping up you never kno:w<br>S3: **<coughs (6)>** | If it is deemed important to indicate the length of the noise (e.g. if a coughing fit disrupts the interaction), this is done by adding the number of seconds in parentheses after the descriptor. |

### D.1.19 NON-VERBAL FEEDBACK

| Example | Description |
|---|---|
| <nods><br><shakes head> | Whenever information about it is available, non-verbal feedback is transcribed as part of the running text and put between pointed brackets . |
| S3: but i think if you structure corporate governance appropriately you can have everything (1)<br>S7: <soft> mhm </soft> **<nods (2)>** | If it is deemed important to indicate the length of the non-verbal feedback, this isdone by adding the number of seconds in parentheses. |

### D.1.20 ANONYMIZATION

A guiding principle of VOICE is sensitivity to the appropriate extent of anonymization. As a general rule, names of people, companies, organizations, institutions, locations, etc. are replaced by aliases and these aliases are put into square brackets **[ ]**. The aliases are numbered consecutively, starting with 1.

| Example | Description |
|---|---|
| S9: that's one of the things (.) that i (1) just wanted to clear out. (2) **[S13]**? | Whenever speakers who are involved in the interaction are addressed or referred to, their names are replaced by their respective speaker IDs. |
| S6: so: (1) ei:ther MYself or mister **[S2/last]** or even boss (.) should be there every year | A speaker's first name is represented by the plain speaker ID in square brackets **[S1]**, etc. |
| S8: so my name is **[S8] [S8/last]** from vienna | A speaker's last name is marked **[S1/last]**, etc. If a speaker's full name is pronounced, the two tags are combined to **[S1] [S1/last]**, etc. |
| S2: that division is headed by (1) **[first name3] [last name3]** (1) | Names of people who are not part of the ongoing interaction are substituted by **[first name1]**, etc. or **[last name1]**, etc. or a combination of both. |
| S5: erm she is currently head of marketing (and) with the **[org2]** (1) | Companies and other organizations need to be anonymized as well. Their names are replaced by **[org1]**, etc. |
| S1: i: i really don't wanna have a: a joint degree e:r with the university of **[place12]** (.) | Names of places, cities, countries, etc. are anonymized when this is deemed relevant in order to protect the speakers' identities and their environment. They are replaced by **[place1]**, etc. |
| S8: he get the <L1cs> diplom {diploma} </L1cs> of **[name1]** university (.) and french university can give him also the <L1cs> diplom {diploma} </L1cs> | Other names or descriptors may be anonymized by **[name1]**, etc., as in e.g. Charles University. |
| S3: erm i- in the **[thing1]** is very well explained. so <2> i can </2> pa- <3> er pass you this </3> th- the definitions.<br>S4: <2> aha </2><br>S4: <3> okay <@> okay </@> </3> | Products or other objects may be anonymized by **[thing1]**, etc. |

### D.1.21 CONTEXTUAL EVENTS

| Example | Description |
|---|---|
| {mobile rings}<br>{S7 enters room}<br>{S2 points at S5}<br>{S4 starts writing on blackboard}<br>{S4 stops writing on blackboard}<br>{S2 gets up and walks to blackboard (7)}<br>{S3 pours coffee (3)}<br>{SS reading quietly (30)}<br>S3: one dollar you get (.) (at) one euro you get one dollar twenty-seven. (.)<br>S4: right. **{S5 gets up to pour some drinks}**<br>S3: right now at this time (3)<br>S1: er page five is the er (4) **{S5 places some cups and glasses on the desk (4)}**<br>S1: i think is the descritip- e:r part of what i have just explained (.) | Contextual information is added between curly brackets **{ }** only if it is relevant to the understanding of the interaction or to the interaction as such. If it is deemed important to indicate the length of the event, this can be done by adding the number of seconds in parentheses.<br>Explanation:<br>The pause in the conversation occurs because of the contextual event. |

### D.1.22 PARALLEL CONVERSATIONS

| Example | Description |
|---|---|
| S1: four billion <spel> u s </spel> dollars. (.)<br>S4: quite impressive (.)<br>S1: er **<to S2>** not quite isn't it **</to S2>** (.) i understand some other countries we handle | To indicate that a speaker is addressing not the whole group but one speaker in particular, the stretch of speech is markedwith (e.g.) **<to S1> </to S1>**, choosing the speaker ID of the addressee. |

| | |
|---|---|
| S7: i've i've found the people very stressed<br>SS: @@@<br>S7: that's (.) i don't know how many of you study here but it's VERY important to push the close the door button in that elevator. this is something i've never <3> seen in sweden </3>**{parallel conversation between S1 and S3 starts}** or anywhere else <4> but it's very important to push this button </4><br>SS: <3> @@@@ </3><br>SS: <4> @@@@@@@@ </4> @@<br>S7: <5> i never even saw this button in another el- elevator </5><br>SS: <5> @@@@@@@@@@ </5> **{parallel conversation between S1 and S3 ends}** @@@ | Wherever two or more conversational threads emerge which are too difficult to transcribe, as a general rule only the main thread of conversation is transcribed. The threads which are not transcribed are treated like a contextual event and indicated between curly brackets **{ }**. |

### D.1.23 UNINTELLIGIBLE SPEECH

| Example | Description |
|---|---|
| S4: we **<un> xxx </un>** for the <7> supreme (.) three </7> possibilities<br>S1: <7> next yeah </7><br>S7: obviously the the PROCESS will **<un>** x **<ipa>** θeɪ■ **</ipa> </un>** (.) w- w- will (.) will take (.) at least de- decade | Unintelligible speech is represented by **x's** approximating syllable number and placed between **<un> </un>** tags.<br>If it is possible to make out some of the sounds uttered, a phonetic transcription of the x's is added between **<ipa> </ipa>** tags. |

### D.1.24 TRANSCRIPTION BORDERS

| Example | Description |
|---|---|
| **beg** CD1_4_00:35> | The **beg**inning of the transcript is noted by indicating the CD number, the track number and the exact position of the respective track in minutes and seconds. |
| **end** CD1_21_01:27> | The **end** of the transcript is noted in the same way. |

| | |
|---|---|
| <end CD1_19_01:27><br>**(gap 00:06:36) {multiple parallel conversations, hardly intelligible}**<br><beg CD1_21_02:03><br><end CD1_24_3:02><br>**(nrec 00:00:45) {change of minidisk}**<br><beg CD2_1_00:00> | A **gap** in the **transcription** is indicated in parentheses, including its length in hh:mm:ss. Curly brackets **{ }** are used in order to specify the reasons for or the circumstances of the gap. An **interruption** in the **recording** is indicated in the same way, but abbreviated as "**nrec**" (i.e. non-recorded). The length you indicate will normally be a guess. |

In addition to the regular mark-up, transcribers supplement the transcripts with Transcriber's Notes in which they provide additional contextual information and observations about other features of the interaction not accounted for in the transcript.

For a detailed discussion of specific aspects of the transcription conventions cf. *VOICE Recording-Methodological Challenges in the Compilation of a Corpus of Spoken ELF* (Breiteneder, Pitzl, Majewski & Klimpfinger 2006)

## D.2 Spelling Conventions

The **VOICE Transcription Conventions** are protected by copyright. Duplication or distribution to any third party of all or any part of the material is not permitted, except that material may be duplicated by you for your personal research use in electronic or print form. Permission for any other use must be obtained from VOICE. Authorship must be acknowledged in all cases.

Version 2.1 June 2007

### D.2.1 CHARACTERS

| Example | Description |
|---|---|
| a b c d e f g h i j k l m n o p q r s t u v w x y z | Only alphabetic **Roman characters** are used in the transcript. No diacritics, umlauts or non-roman characters are permitted in the running text. |

### D.2.2 DECAPITALIZATION

| Example | Description |
|---|---|
| S8: so you really can <@> control my **e**nglish </@> | **No capital letters** are used except for marking emphasis (cf. mark-up conventions). |

### D.2.3 BRITISH SPELLING

| Example | Description |
|---|---|
| British spelling | British English spelling is used to represent naturally occurring ELF speech.<br>The Oxford Advanced Learner's Dictionary (**OALD**), **th edition**, is used as the primary source of reference.<br>If an entry gives more than one spelling variant of a word, the first variant is chosen. If there are two separate entries for British and American spelling, the British entry is selected. |

### D.2.4 SPELLING EXCEPTIONS

| Example | Description |
|---|---|
| cent**er**, theat**er**<br>behavi**or**, col**or**, fav**or**, lab**or**, neighb**or**<br>defen**s**e, offen**s**e<br>dis**k**<br>progra**m**<br>travel (-l-: trave**l**ed, trave**l**er, trave**l**ing)<br>S2: we are NOT quite sure if it will REALLY be (.) privati**z**ed next year | The 12 words listed on the left **and all their derivatives** are spelled according to American English conventions (e.g. colors, colorful, colored, to color, favorite, favorable, to favor, in favor of, etc.).<br><br>In addition, **all words** which can be spelled using either an -is or an **-iz** morpheme are spelled with **-iz**(e.g. to emphasize, organizations, realization, recognized, etc.). |

### D.2.5 NON-ENGLISH WORDS

| Example | Description |
|---|---|

| | |
|---|---|
| S1: <L1de> wieso **oe**sterreich? {why austria} </L1de><br>S3: <LNfr> c'est ferm**e**? {is it closed} </LNfr> | Non-English words are rendered in the standard variant of the original language (i.e. no non-standard dialect). The roman alphabet is always used, also in the case of languages like Arabic or Japanese.<br>**No umlauts** (e.g. NOT österreich), **no diacritics** (e.g. NOT fermé) and **no non-roman characters** are permitted. |

### D.2.6 FULL REPRESENTATION OF WORDS

| Example | Description |
|---|---|
| S7: the students that (.) decide freely to enter (.) this kind of master knows (.) for example that **he** can (.) at the end achieve (.) sixty credits | Although words may not be fully pronounced or may be pronounced with a foreign accent, they are generally represented in standard orthographic form.<br>Explanation:<br>S7 is Italian and pronounces the he in he can as /ɪ/, swallowing the initial h. Nevertheless, this is regarded as a minor instance of L1 accent and therefore represented in standard orthography (he). |

### D.2.7 FULL REPRESENTATION OF

| Example | Description |
|---|---|
| oh/zero, two, three, ... one hundred, nineteen ten, eighteen twenty-seven, ... | Numbers are fully spelled out as whole words. British English hyphenation rules apply. |
| missis (for Mrs), mister, miss, mis (for Ms), doctor, professor, ... | Titles and terms of address are fully spelled out. |
| et cetera, saint thomas, okay,... | Forms that are usually abbreviated in writing, but spoken as complete words are fully spelled out. |

### D.2.8 LEXICALIZED REDUCED FORMS

| Example | Description |
|---|---|
| cos<br>gonna, gotta, wanna | Lexicalized phonological reductions are limited to the four on the left.<br>All other non-standard forms are fully spelled out (e.g. /hæftə/ = have to). |

### D.2.9 CONTRACTIONS

| Example | Description |
|---|---|
| i'm, there're, how's peter, running's fun, ...<br>i've, they've, it's got, we'd been, ...<br>tom'll be there, he'd go for the first, ...<br>we aren't, i won't, he doesn't, ...<br>what's it mean, where's she live, how's that sound ...<br>let's | Whenever they are uttered, all standard contractions are rendered.<br>This refers to verb contractions with be (am, isare), have (have, has, had), will and would as well as not-contractions.<br>Additionally, 's is used to represent does when reduced and attached to a wh-word. It is also used to represent the pronoun us in the contracted form let's. |

### D.2.10 HYPHENS

| Example | Description |
|---|---|
| S3: more than thirteen years of experience er working in (.) er (.) design and development (.) er of (1) **real-time** software (.) er for industrial (.) implications<br>S2: we would allow that within er an international **cooperation** (.) | Hyphens are used according to British English hyphenation rules. The OALD, 7th edition, is used as the primary source of reference.<br>If an entry gives more than one spelling variant of a word, the first variant is chosen. |

### D.2.11 ACRONYMS

| Example | Description |
|---|---|
| S10: for the development of joint programmes within the **unica** networks. | Acronyms (i.e. abbreviations spoken as one word) are transcribed like words. They are not highlighted in any way. |

### D.2.12 DISCOURSE MARKERS

All discourse markers are represented in orthography as shown below. The lists provided are closed lists. The items in the lists are standardized and may not represent the exact sound patterns of the actual discourse markers uttered.

| Example | Description |
|---|---|
| yes, yeah, yah | **Backchannels and positive minimal feedback** |
| okay, okey-dokey | |
| mhm, hm | (closed sound-acknowledgement token) |
| aha, uhu | (open sound-acknowledgement token) |
| no | **Negative minimal feedback** |
| n-n, uh-uh | |
| er, erm | Hesitation/filler |
| huh | tag-question |
| yay, yipee, whoohoo, mm: | **Exclamations** |
| | joy/enthusiasm |
| haeh | questioning/doubt/disbelief |
| a:h, o:h, wow, poah | astonishment/surprise |
| oops | apology |
| ooph | exhaustion |
| ts, pf | disregard/dismissal/contempt |
| ouch, ow | pain |
| sh, psh | requesting silence |
| oh-oh:, u:h | anticipating trouble |
| ur | disapproval/disgust |
| oow | pity, disappointment |

| | |
|---|---|
| S3: <L1ja> **he:** </L1ja><br>SX-m: <L1de> **ach ja {oh yes}** </L1de> | What are clearly L1-specific discourse markers are marked as foreign words. Due to the wide range of these phenomena in different languages, the L1-list is open-ended.<br>A translation is added whenever this is possible. |

For a detailed discussion of specific aspects of the transcription conventions cf. *VOICE Recording-Methodological Challenges in the Compilation of a Corpus of Spoken ELF* (Breiteneder, Pitzl, Majewski & Klimpfinger 2006)

# E German Summary
# Deutschsprachige Zusammenfassung

Die vorliegende Arbeit behandelt den Aufbau und die informationstechnische Umsetzung einer sprachwissenschaftlichen Ressource zur Erforschung von Englisch als Lingua Franca. In den einzelnen Kapiteln werden verschiedene Schwerpunkte gesetzt, die ein breites Spektrum von konzeptuellen und theoretischen Überlegungen bis hin zu Aspekten der direkten Umsetzung abdeckt.

Die Hauptschwerpunkte der Arbeit liegen in einer grundlegenden Betrachtung sprachwissenschaftlicher Annotationssysteme (siehe *2. Annotation p.4*), der Beschreibung der formalen Struktur und der Entstehung des VOICE Transkriptionssystems, eines Überblicks über zusätzlich erforderliche Daten (d.h. Meta-Daten) bis hin zu einer Diskussion des verwendeten Dokumentenformates.

In der Arbeit wird in den einzelnen Teilbereichen immer wieder auf das Spannungsfeld zwischen dem wissenschaftlich Wünschenswerten und dem informationstechnisch Machbaren hingewiesen. Das beschriebene Transkriptionssystem vereint kognitive und formale Aspekte in einer Auszeichnungssprache um ein geeignetes Eingabeformat bereit zu stellen. Die darauf aufbauenden Formate, wie zum Beispiel das Korpusformat in XML, sind in dieser Arbeit konzeptionell und in wichtigen Implementationsdetails dokumentiert. In diesem Sinne ist der Text auch als Dokumentation für eine technische Annäherung an die Korpusresource VOICE zu lesen.

# Lebenslauf (akademisch)

## Persönliche Daten

Name: Stefan Majewski

## Ausbildung

### *Studium*

| | |
|---|---|
| 03/2008 - 09/2011 | Studium der Anglistik (mit Schwerpunkten aus der Informatik und Soziologie) Abschlussarbeit: "Design and Implementation of a Research Infrastructure for a Corpus of Spoken ELF" |
| 03/2001 - 02/2008 | Studium der Soziologie und der Anglistik |
| 10/1999 - 02/2002 | Studium der Elektrotechnik |

### *Weiterbildungen und Praktika*

| | |
|---|---|
| 03/2011 | Teilnahme am XML Datenbanken Workshop im Rahmen der XML-Prague |
| 11/2010 | Teilnahme am ODD Konferenzworkshop im Rahmen der TEI Konferenz Zadar |
| 03/2010 | Teilnahme am eXist Pre-Conference Workshop im Rahmen der XML-Prague |
| 02/2007 | Teilnahme am TEI Training im Oxford University Computing Service |
| 07/1999 - 09/1999 | Elektrotechnisches Grundpraktikum bei Siemens München |

### *Schulbildung*

| | |
|---|---|
| 09/1989 - 06/1998 | Gymnasium Gauting: Abschluss Abitur |

## Publikationen und Präsentationen

### *Publikationen*

Majewski, Stefan. (forthc.). "Textbasierte Transkription in VOICE". *Tagungsband des Zentrums für Sprachwissenschaft, Bild- und Tondokumentation der österreichischen Akademie der Wissenschaften zum Thema "Transkriptionssysteme: Sprache - Bild - Ton. Codierung gesprochener Sprache"*. Wien: Praesens.

Breiteneder, Angelika; Klimpfinger, Theresa; Majewski, Stefan; Pitzl, Marie-Luise. 2009. "The Vienna-Oxford International Corpus of English (VOICE) - A linguistic resource for exploring English as a lingua franca". *ÖGAI-Journal* 28/1, 21-26.

Breiteneder, Angelika; Pitzl, Marie-Luise; Majewski, Stefan; Klimpfinger, Theresa. 2006. "VOICE recording - Methodological challenges in the compilation of a corpus of spoken ELF". *Nordic Journal of English Studies* 5/2, 161-188. (Available at http://hdl.handle.net/2077/3153)

Kastovsky, Dieter; Rieder, Angelika; Weiss, Corinna; Majewski, Stefan. 2002. *Sprachlehrforschung in Österreich: Stand und Perspektiven*. Wien: BMBWK.

## Publikationen und Präsentationen (fortgesetzt)

### *Präsentationen*

Majewski, Stefan; Wang, Lixun. (forthc.). 'Bringing VOICE and ELFiA together. An exercise in collaborative corpus creation'. Symposium on 'The VOICEs of Europe and Asia: diversity of data but harmony in approach', convened by Barbara Seidlhofer and Andy Kirkpatrick. 16th World Congress of Applied Linguistics (AILA 2011), Beijing, 23-28 August 2011.

Majewski, Stefan. 'VOICE XML: From Transcript to XML Corpus'. Individual Paper, 4th International Conference of English as a Lingua Franca, Hong Kong, 26-28 May 2011.

Majewski, Stefan. 'VOICE Online, bringing VOICE to the web?'. Visual Presentation, 3rd International Conference of English as a Lingua Franca, Vienna, 22-25 May 2010.

Majewski, Stefan. 'CorpusQuery, the software behind VOICE Online'. Individual software presentation, ICAME 31, Giessen, 26-30 May 2010.

Klimpfinger,Theresa; Majewski, Stefan: 'VoiceScribe: Transkriptionssoftware für die Codierung von Englisch als Lingua Franca'. Invited lecture, first annual meeting of the Centre of Linguistics and Audiovisual Documentation (LAVD) at Austrian Academy of Science (ÖAW) 'Sprache-Ton-Bild. Codierung gesprochener Sprache', Vienna, 29 January 2009.

Breiteneder, Angelika; Klimpfinger,Theresa; Majewski, Stefan; Pitzl, Marie-Luise: 'VOICE - eine sprachwissenschaftliche Ressource zur Erforschung von Englisch als Lingua Franca'. Invited presentation, VERBAL Workshop CLARIN (Common Language Resources and Technology Infrastructure), Vienna, 5 December 2008.

### *Software*

Majewski, Stefan. 2011. *Zotero Export Plugin: tei-zotero-translator*. (beta) http://code.google.com/p/tei-zotero-translator.

Majewski, Stefan. 2007. *VoiceScribe*. (version 1.0.2). http://www.univie.ac.at/voice/page/voicescribe.

## Berufliche Erfahrung

| | |
|---|---|
| 06/2005 - | Research Assistant(IT) im FWF-finanzierten Projekt „Vienna-Oxford International Corpus of English (VOICE)" http://www.univie.ac.at/voice<br>Aufgaben:<br>- Planung und Umsetzung der Softwarearchitekturen und der erforderlichen Systemkomponenten<br>- Koordination mit Open-Source Projekten und Experten aus Fachgemeinschaften (z.B. TEI, eXist)<br>- Vorstellung der technischen Aspekte auf Konferenzen und in Publikationen<br>- sonstige IT-Agenden des Projekts |
| 05/2002 - 07/2002 | Mitwirkung an der Umsetzung und Auswertung einer Studie des bmbwk zur Entwicklung der Sprachlehrforschung in Österreich |