MATE Deliverable D2.1
MATE Dialogue Annotation Guidelines

III. Annexes

Annex 1: General markup guidelines

Andreas Mengel

This chapter deals with general considerations useful for the formal encoding of linguistic annotation data.

1. Markup

As the term *markup* itself is often used in a very broad sense, the following three sub-concepts that are covered should be distinguished:

- the **phenomena** investigated: Some researchers will describe the words and their properties, some will go for the relations of chunks of words, and others investigate their sound structures;

the **theory**: This is how the linguistic phenomena are labelled, they may be called *words* and the theory might claim that these can be put into part of speech categories like A, B or C, or another theory might say that the POS values are NN, OO, PP and QQ. And perhaps these categories are not called *POS* but *wc* (*word categories*);

- the **markup**: This is the kind of characters, the grammar, the formalism used to represent the labels used by a given theory for the description of linguistic material. So if a piece of the corpus is:

  "*I saw it cross the street*"

  The individual words could be marked up in these two ways:

`I saw it cross the street`	normal orthographic markup
`<w>I</w><w>saw</w><w>it</w><w>cross</w><w>the</w><w>street</w>`	XML

These three concepts (*phenomena*, *theory*, *markup*) are orthogonal. So one can have XML, SGML, or xwaves/*xlabel* as the markup; a Chomskyan or Tesnierian approach to syntax; and one can have descriptions of the intonation, syntax, or semantics; and any possible combination of the three.

2. Data files

There are various types of files specifying different kinds of information to be encoded during the annotation of corpora entities:

non-XML data
• source data: these are data that have been produced to record linguistic data and prior to any annotation process

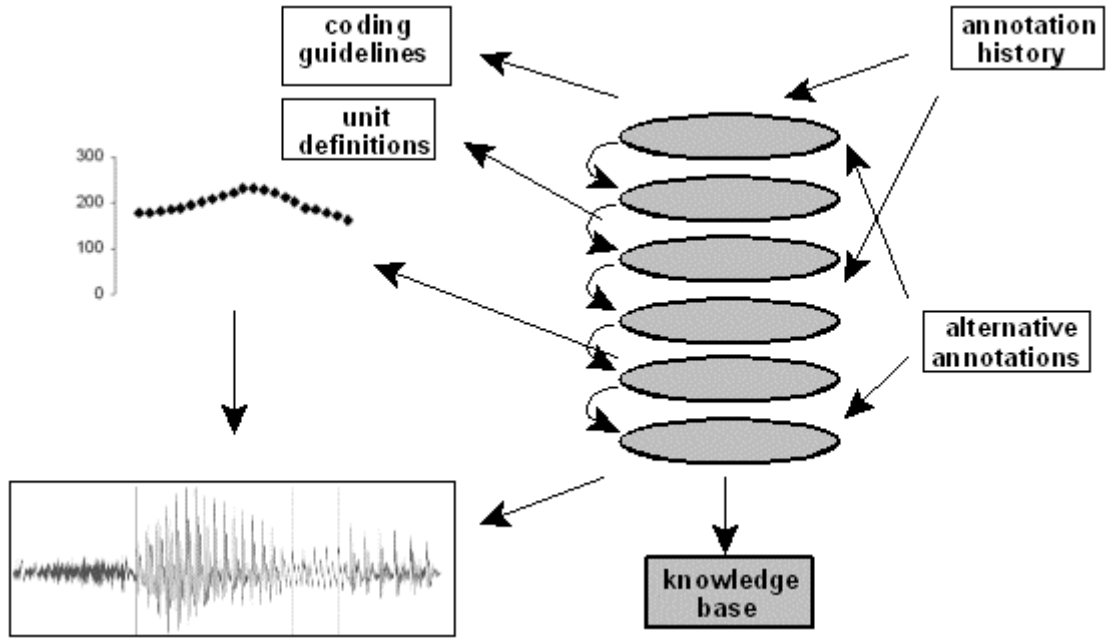
- text: text data are transcripts of speech or other written documents
- non-text: non-text data are binary data that represent recordings of speech behaviour
  - audio: audio data may be speech files
  - graphical: graphical representations of speech behaviour or the communication situation
    - pictures: pictures taken during a communication situation
    - video: visual recordings of the communication situation

(possibly) XML data

- analysis data: mathematical analyses performed on any of the data
  - signal measurement data: evaluations of the speech signal, e.g. f0 and other spectral analysis
  - statistical calculations: descriptive statistical analysis and statistical tests on any of the data
- annotation data: data that have been produced to categorize linguistic phenomena
  - level annotation: annotation of individual linguistic levels of description
- knowledge resources: data that describe phenomena in a generic way and relate to phenomena the properties of which are considered stable during the linguistic situation annotated
  - lexical data: type information provided for individual words
  - universe data: information on the situation and objects in a given communication situation
- annotation process data: data that relate to the process of annotation
  - prescriptive information: these are data that are a reference for the coder when annotating
    - unit definitions: definitions of individual units to segment and categorize
    - coding guidelines: descriptions on the procedure of annotation dependant on the linguistic level
  - descriptive information: this is information related to the actual annotation
    - source data list: a list of data that have been used as source data
    - log files: information on actual annotation processes such as the creation, modification or deletion of a tag
    - settings: these are customizations of the software environment
      - user privileges: settings that describe permissions of the user for access and modification of data
      - graphical user interface data: general settings of the software environment
      - display style sheets: specific display and access descriptions for

	given annotation tasks ▪ user preference data: user controlled software environment settings

The following figure represents these types of information and relations among them. The ellipse in the middle symbolize annotation levels. Square boxes are non-level information. Arrows indicate referring relations: Arrows between levels denote the reference between levels, e.g. reference from the word level to the phone-level; arrows from the level files to square boxes symbolize reference to rules that have been used during the annotation; arrows pointing from square boxes to ellipses indicate (meta-)annotation of the level label files. The shading of boxes or ellipses denotes that their contents is annotation of linguistic objects. Note that the direction of the arrows is also a reflex of the process of annotation, i.e. in most cases those objects pointed to are produced earlier.



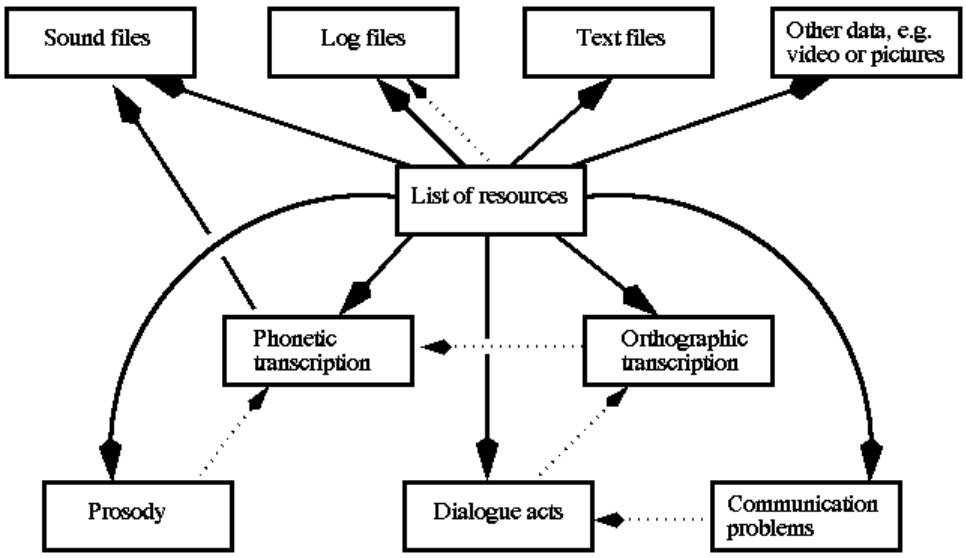
The data produced during the annotation of a dialogue will be kept in separate files, but it should also be ensured that their mutual relationship and the fact that they belong to the same annotation project is transparent. There are two strategies to specify this membership information:

- **list**: A project resource file is established that keeps the file names of the data used for the annotation project (annotation process information)
- **links**: Relations are established between elements of different levels of description; the elements refer to each other and are located in separate files (level annotation information)

In general, *links* between any two types of documents can be established by the means described in the table below.

from	to	by means of
list of resources	files used during annotation	file name attribute
XML document	non-XML document	file name attribute
non-XML document	any other document	not possible
XML document	XML document	href attributes between elements

The following figure represents the different types of files and explicit references. It is an adapted version of a figure in D1.2 which has been modified in order to also reflect the direction of reference. Bold lines represent reference by file name, dotted lines represent reference (href-attributes) between annotation elements of different level annotation files.



3 Markup Conventions

This section is dedicated to all markup aspects that can be described on a general level and serve to be applied to level specific markup descriptions. The following is a general guideline for the mapping of XML syntax onto linguistic concepts. In order to make this mapping as uniform, efficient, and consistent as possible these guidelines discuss some of the problems encountered and offers proposals for solutions.

XML has been chosen to be used for the representation of annotation data in the MATE project. Besides the actual good support of software for this standard, there are the following reasons for XML as the representation model:

XML uses

- one general description model containing
    - elements
    - their attributes
    - part-whole relations betwen elements

    - other relations between elements

- one encoding format to represent, relate, and distinguish these kinds of information

This is the reason why XML can be assumed to be able to encode markup irrespective of the phenomena described or theoretical approach taken. The underlying (object oriented like) model of description XML is based on can be assumed by almost all theories: There are phenomena (elements) some of which can be split into sub-phenomena (embedded elements), these phenomena have relations and properties (attributes).

On the formal side this model is reflected by a uniform syntax described above: XML-elements are enclosed by angular brackets

```
<xyz att="val">
```

inside which the values (`val`) of attributes (`att`) are specified: In XML **Elements** are those entities that conceptually group together descriptions of linguistic items. In XML element names are put after the <, such as in `<word>`. **Attributes** are those entities that allow further description of entities by specifying property dispositions. Attribute names denote the property dimension, its actual status for a given element is described by an attribute **value**. In `<word pos="NN">` `pos` is the attribute, the exact value is `NN`. **PCDATA** are any characters which are not included in a pair of one opening (<) and one closing (>)angular bracket (see below).

Despite this general approach, XML does not provide the following:

- typed/grammar specification of attribute values for the distinction of floats, characters or for the definition of attribute values by regular expressions or BNF grammars,

- inference models for element values that allow for centralized specification of properties that are shared by more than one element,

- applicability restriction of attributes that are mutual exclusive e.g. words that are nouns cannot have tempus information and case information is not applicable to verbs.

The XML community is aware of these problems and proposals are under way to improve and extend the XML standard. The inference problem is discussed in more detail in the chapter on cross-level annotation.

3.1 Minimal redundancy

The markup produced should be minimally redundant. That means that any information applying to more elements that conceptually depend on each other should only be represented once in the document if it is possible to find general means to infer the information marked at one element when accessing the other. On the level of defining elements, attributes, and values for concepts to be annotated, this principle has the effect that only those attributes and values that cannot be inferred from the element name or the values of other attributes of the element, have to be specified. To give an example: In the case of segmenting speech into phones, one would not have to specify the voicing of sounds as extra attributes once the identity of a sound is determined, thus

redund.xml

```
<phone type="a:" voiced="true"/>
```

is only sensible if the theory or the application assumes that there are vowels which can be voiceless (e.g. in the case of whispering).

In general, elements used for tagging should not carry the theory itself but that part of information that cannot be predicted only.

3.2 Maximal consistency

If rules for the mutual dependency of information represented at different places of a document can be stated, this type of consistency should be enhanced. One means of enhancement is the minimal redundancy principle, as information placed only at one place that that can be inferred from some other place of the document will only have to be updated once. Furthermore *maximal consistency* covers the area of reference for the storage of annotation: If the structure of the tags of the items to be linked varies in an unpredictable fashion across corpora or parts of corpora, no reliable (automatic) tagging or retrieval can be guaranteed. A further prerequisite of level annotation is the existence of one general model that can be applied to any kind of tagging, i.e. there is a need for a minimal standard of tagging on all levels to be labelled.

3.3 Universal parsability

The markup used should be universally parsable. This has three levels of consequence.

- First, a general grammar should be used, supporting a uniform representation of different entity types. One example of this is XML which guarantees that any linguistic information can be parsed by one type of parser once it is encoded in XML.
- The second level of *parsability* refers to the actions or the meaning of the markup. This is the behaviour of a piece of software after having parsed the file. Examples of this behaviour may be the representation, display, and reactions to user input. This second level of parsability cannot be encoded my XML, it has to be defined elsewhere.
- A third level of *parsability* is the processing of information (stored in XML, processed and displayed by the computer) by human annotators: The information displayed must be accessible for them, too. Thus, additional information is needed to represent the actual meaning and theory behind the XML annotation applied to linguistic data.

3.4 Optimal maintainability

Typical applications of XML are hierarchies of different elements which are nested. It is not possible however to design one general hierarchical model in which all linguistic information to be described in speech can be represented. It is easy for elements like sounds, words, phrases and sentences, but is not in the case of sounds, pauses, background noise, head turns,

and co-reference. Entities of these different categories have no conceptual dependencies which could be represented in hierarchical structures. This is the reason why the tagging of conceptually or theoretically exclusive levels must be put into different XML files. Yet, the encoding of mutual theoretically dependent information should also reside in separate files, as they will ideally be produced at different occasions and element-type wise. If one has to add a higher level of annotation to an existing lower level tagging file, this would mean that the file has to be altered requiring complex manipulations of the file. Thus, each conceptually different level of description should be placed in a file of its own.

3.5 Naming conventions

In general all names of elements, attributes, and values should be in lower case only. This looks like a layout fashion but makes reading and style of documents more consistent. Also - where possible - names used for elements, attributes and values should consist of more than one or two characters. In the case where names for elements proposed by the TEI guidelines are used, the names used should be employed, although <u> and <w> are not favourable as they are not very intuitive for people not knowing the TEI guidelines.

3.6 Linking information

In general, there should be the possibility to link and to align various levels of description. For the sake of the next example assume it is a sentence and a word markup. Suppose, there is a word tagger that provides the user with basic tagging of words which results in a document like:

word.xml

```
<w id="w_01">take</w>
<w id="w_02">this</w>
<w id="w_03">example</w>
```

Adding annotation of the sentence level would either

- a) produce a (new) document which is a copy of the first one (or a new version of the first document) plus sentence tags (<s>) added

word2.xml

```
<s>
  <w id="w_01">take</w>
  <w id="w_02">this</w>
  <w id="w_03">example</w>
</s>
```

OR

- b) be a second file with <s> elements that hyperlink to the first one

sent.xml

```
<s href="word.xml#id(w_01)..(w_03)"/>
```

As stated above, the second approach is recommended which - depending on the DTD - treats the <w> elements as children of <s> elements like in variant a), but in a non-invasive fashion [see [1]].

3.7 PCDATA

PCDATA are all textual entities which are not inside any element, i.e. outside angular brackets (<>). In the case of orthographic text that shall be marked up and integrated into a corpus of dialogue annotation, words will be the basic objects of markup. Around each word - if marking up words is the application - there would be an element start tag <w> and an element end tag </w>:

word.xml

```
<w id="w_001">These</w>
<w id="w_002">are</w>
<w id="w_003">the</w>
<w id="w_004">words</w>
```

In this case, the elements are filled by PCDATA which we perceive as orthographical words. Sentence annotation building upon this would add <s> and </s> around the text before:

sentence.xml

```
<s id="s_001">
  <w id="w_001">These</w>
  <w id="w_002">are</w>
  <w id="w_003">the</w>
  <w id="w_004">words</w>
</s>
```

In this case, the <s> element is filled, too.

Empty elements are those which do not include neither other elements nore PCDATA, e.g. in the case of sentence annotation that refers to other elements by an `href` attribute:

sentence2.xml

```
<s id="s_001" href="word.xml#id(w_001)..id(w_004)"/>
```

It is recommended to use PCDATA only if these PCDATA are textual information (marked up text from source data). The use of PCDATA inside an element for specification of values is only preferable if these are very long and explicit.

In the following two examples there are examples where location information of a situation is provided. The second example is a case for choosing PCDATA.

situation1.xml

```
<situation id="sit_0223" place="home"/>
```

situation2.xml

```
<situation id="sit_0223">
  <place id="loc_001">
    The participants are sitting in the living room
    of the apartment of the speaker named Martha.
  </place>
</situation>
```

All other information should be coded by attribute values, links and embedded elements.

3.8 Elements vs. Attributes

In general when annotating speech data, elements are often entities that have an extension in time. For many categories of speech phenomena, there are not only labels but also notation systems, i.e. sets of symbols that denote the item as such and its category (cf. ToBI, POS). When describing linguistic levels, one has to decide if the standard labels will be used as attribute values or as elements in the markup [2].

book.xml

```
<book title="The Call of the Wild" author="London, Jack"/>

                    or

<book author="London, Jack">The Call of the Wild</book>

                    or

<book>
  <title>The Call of the Wild</title>
  <author>London, Jack</author>
</book>
```

Or, see the following options for representing prosody and phrase types:

Phrases:

phrase.xml

```
<phrase type="NP"/>
<phrase type="VP"/>
<phrase type="NP"/>

        or

<NP/>
<VP/>
<NP/>
```

ToBI labels:

notobi.xml

```
<pros type="L*"/>
<pros type="H*"/>
<pros type="L*H"/>

        or

<L*/>
```

```
<H*/>
<L*H/>
```

As a matter of convention, one should choose that level of abstraction that allows to segment a series of entities, name each of these entities by that one term which can be applied to all of them and encode their differences by attributes and values respectively. Note, that there is an interrelation between mutual exclusive attribute values and the choice of level of abstraction: If the description level and element type chosen is `<w>` a part-of-speech value of "`noun`" theoretically blocks the application of tense. If the element types chosen were `<noun>`, `<verb>`, `<adj>`, etc., this would certainly not happen, but the very information that all of these elements belong to one group of phenomena will be lost and is not exploitable for query access to the data. One further solution is the additional encoding of abstraction level information.

tobi.xml

```
<s>
  <w>
    <det num="sg" case="nom">The</det>
  </w>
  <w>
    <noun num="sg" case="nom">tree</noun>
  </w>
  <w>
    <verb num="sg" tense="past">grew</verb>
  </w>
</s>
```

Attribute values should be chosen in a way that allows as much conceptualization as possible. Consider phonetic sounds: In the case where speech has to be segmented into sounds, one could think of a set of attributes specifying articulatory or auditive properties of these vowels as attributes, the values of which are set to plus (+) or minus (-). The alternative would obviously be to use the conventional IPA or SAM-PA symbols as values of a single attribute, e.g. "`type`". The first alternative has two disadvantages: If one assumes that the set of possible combinations of articulatory or auditive properties is known and the sounds to be segmented are limited to a small subset of all possible combinations, it seems an effort too hard to choose this option. The other reason is that there might be mutual dependencies of the values e.g. no plosive can be rounded. These dependencies cannot be constrained automatically by the grammar of a DTD or so. Thus it is highly recommended to choose that set of attributes that guarantees maximal mutual independence of the attributes, i.e. find entity descriptions which are used to encode typical attribute-value constellations, in this case sound symbols. It might not always be possible to find a set attributes that are not mutually dependent, cf. the word attribute CASE which is not applicable to verbs as discussed above.

3.10 Time information

Since speech is a behaviour, behaviour is an action and action involves the concept of time, time is an obviously important property of speech units. In order to assess speech aspects like synchronicity, the sequence or the duration of speech events etc. are important units to be described. And as time information is the minimal chain of common reference across levels, time description conventions should be standardized to the maximum.

There are various options for the encoding of this information:

- First of all, one has to decide between specifying time in samples (signal measurement points) or seconds/miliseconds: The advantage of using samples is that samples are the finest grains available relative to the sample frequency used for a given segment of recorded speech. The disadvantage of using samples is that it is more difficult to compare time relations across documents: In order to access time, the sample frequency would have to be available for its calculation.

- Second, the properties - and attributes respectively - to use have to be chosen. In principle one could either employ `start` and `end` or `duration`. As `duration` can be calculated by the other two properties, but not the other way around, `start` and `end` seem more appropriate and are recommended as standard attributes. Yet, for some elements this concept seems difficult to apply, e.g. in the case of f0 values, door slams or other events which conceptually do not seem to have an extension in time. However, to be consistent, these elements should have the same attributes with the specialty that the values of the `start` and the `end` attribute are equal.

- Third, one could argue to leave one of the attributes out because many times, the start time of an element equals the end time of the element before. Unfortunately, this is not always the case, e.g., if there are pauses, utterances of other speakers, or simply because the element is located at the beginning or the end of the event chain. Some of these problems could only be solved if one would have all physical events listed and annotated in one document, what would require very complex DTDs and massive efforts in the handling of information. As it seems easier to keep elements of different conceptual levels in separate files, and it would be more complicated for the user or the software to decide when to include this information and when not, `start` and `end` attribute should always appear ensemble in an element tag.

- The fourth option touches the completeness of time information for every element on every level or whether time information could be inherited. For time information, one would want to provide `start` and `end` on only one layer and make elements from higher levels of description (e.g., words) that are conceptually related to these units (e.g., phones) point to that information or inherit it. Basically two options can be considered:

    - There is a special kind of attribute that allows to inherit values of other attributes, such that one can say that the value of the `startinh` attribute of YZ elements inherits that information from the `start` attributes of XY elements:

yz.xml

```
<xy id="xy001" start="000" end="002"/>
<xy id="xy002" start="002" end="005"/>
<xy id="xy003" start="005" end="009"/>
```

xy.xml

```
<yz startinh="xy001.start"/>
```

    - There is no explicit `start` or `end` attribute in higher level elements at all but for the sake of getting this information, a processor has to go down in the element hierarchy and check all first children (of first children)* until this information is found specified somewhere. Exactly the same procedure is used

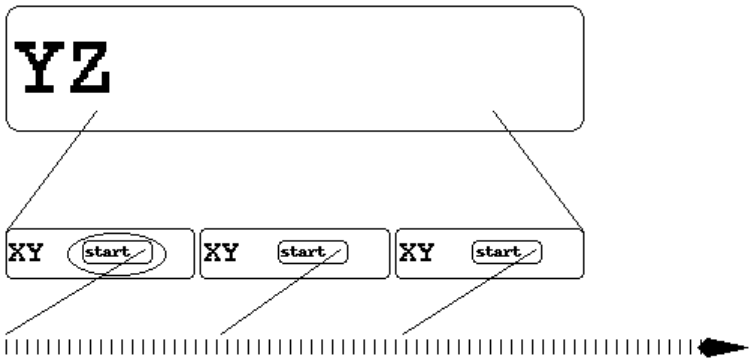
for the end value, with the only difference that one uses the last children (of children)* in that case.

xy.xml

```
<xy id="xy001" start="000" end="002"/>
<xy id="xy002" start="002" end="005"/>
<xy id="xy003" start="005" end="009"/>
```

yz.xml

```
<yz href="xy.xml#id(xy001)..id(xy003)"/>
```



The second variant is recommended: If two levels of description are situated in the same hierarchy and the elements of one of them are parent elements of the other, then it is proposed to note start and end information for every element of the lower or lowest level in that conceptual hierarchy. The attributes start and end of the higher level elements can be calculated by evaluating the start value of the first embedded element and the end value of the last embedded element [2]. This in accordance with the two principles stated above, namely minimal redundancy and maximal consistency. If the time information is changed and all time information were kept separately, i.e. put to every level individually, then information of each element on each level would have to be changed. If there is only one basic representation all other tags refer to, and time information of these tags is inferred form the basic tag, then the time information has only to be changed once and changes applied to the time values on the lowest level will automatically affect the time specification of the associated higher levels.

In the following example, the start value of the sentence (<s>) would be 0.01 and the end value would be 0.62.

time.xml

```
<s id="s_001">
  <w id="w_001" start="0.01" end="0.20">It</w>
  <w id="w_002" start="0.20" end="0.37">was</w>
  <w id="w_003" start="0.37" end="0.42">time</w>
  <w id="w_004" start="0.42" end="0.62">again</w>
</s>
```

To exploit this principle most effectively, time information and thus the initial transcription of spoken material should be applied to the level with the smallest units under investigation (cf. the section on cross-level).

Time is relative and if an element has the `start` value of `4.34` which is to be compared to the `start` value of other elements in another hierarchy, this may cause problems as this time information will in both cases refer to the time elapsed relative to the beginning of the document they reside in only but may be incompatible for many cases. It is recommended that time is either specified relative to midnight, January 1, 1970 UTC (universal time code) or relative to the beginning of the recording. For the latter case it is useful to apply the special attribute `rectime` that states the time distance of the beginning of the recording relative to midnight, January 1, 1970 UTC. Even if a software is not able to calculate and compare the time information of elements of different documents, it will be easy to check if the documents are compatible, and thus if a query that compares `start` and `end` information of these documents is sensible.

References

[1] Isard, A., McKelvie, D. and Thompson, H.S.: Towards a Minimal Standard for Dialogue Transcripts: A New Sgml Architecture for the HCRC Map Task Corpus. Proceedings of the 5th International Conference on Spoken Language Processing, ICSLP98, Sydney. http://www.cogsci.ed.ac.uk/~dmck/Papers/icslp98.ps

[2] The SGML/XML Web Page: SGML/XML Elements versus Attributes. http://www.oasis-open.org/cover/elementsAndAttrs.html

Annex 2: Coding Modules

The concept of *coding modules,* i.e. sets of descriptions for the annotation of linguistic phenomena on a given level covers the set of these data (D1.2). *Coding module* data are a central means of communication between the

- developer of coding guidelines,
- the coder of linguistic data, and
- the user of the annotated data

and are seen as vital for the effective and reusable annotation of linguistic events.

Prosody

1. Phonetic Transcription Coding Module

- name: Phonetic Transcription
- coding purpose: segmentation of speech into phonetically labeled segments (SAMPA scheme)
- coding level: Prosody
- data sources: spoken corpora (speech files, orthographic transcription)
- module references: orthographic transcription module (optional)
- description: The level defines a base element, the <phone> element, corresponding to a segment in the speech signal, labeled according to its phonetic features. A <syllable> element may be added, consisting of a sequence of <phones>. The annotation at this level is a transcription and a segmentation, in the sense that it refers directly to the speech signal, recognizes the uttered sounds and splits the speech continuum into phonetic chunks. Each <phone> will then be classified with a phonetic label and associated with time information specifying its start and end instants. Higher linguistic levels, like the phonological prosodic levels or the orthographic word level, might inherit time information from the phonetic level by linking their elements with <phone> elements or <syllable> elements.

  The scheme adopted here for phonetic transcription is SAMPA [Wells et al., 1992], which is intended for multi-lingual phonetic transcription. In the original SAMPA notation, a transcription is a stream of phonetic labels and diacritics, where labels classify phones and diacritics give further specifications about phones, with the exception of stress marks which implicitly refer to the following syllable. In our adaptation, the <syllable> element is made explicit as a second layer built on top of the <phone> layer.
- example: The following example shows the phonetic transcription of the Spanish word *'casa'* ('house') and its corresponding syllabic segmentation, using the <phone> and <syllable> elements:

phone.xml

```
<phone id="phn_01" type="k" start="345" end="390"/>
<phone id="phn_02" type="a" start="390" end="450"/>
<phone id="phn_03" type="s" start="450" end="490"/>
<phone id="phn_04" type="a" start="490" end="540"/>
```

syllable.xml

```
<syllable id="sllbl_01" stress="&quot" href="phone.xml#id(phn_001)..id(phn_002)"/>
<syllable id="sllbl_02"               href="phone.xml#id(phn_003)..id(phn_004)"/>
```

- markup declaration:

```
ELEMENT <phone>

ATTRIBUTES:
id [ASCII]
type [ASCII]*
start [FLOAT]
end [FLOAT]
```

* The attribute 'type', although defined as ASCII data, can only contain an allowable (language-dependent) combination of SAMPA symbols and diacritics.

```
ELEMENT <syllable>

ATTRIBUTES:

id [ASCII]
stress &quot, %
start [FLOAT]
end [FLOAT]
```

- coding procedure:
  - Manual phonetic segmentation:
    - select the speech file and open the synchronized windows for phonetic segmentation and waveform and spectrum display
    - zoom until a detailed inspection of the signal is possible
    - inspect and listen to the signal portion until the uttered phonemes are recognized
    - select a phonetic label for the first phone
    - identify its boundaries according to the segmentation criteria and mark them by placing the cursor on the proper point on the time-axis (this should automatically set the time attribute)
    - after phonetic segmentation is concluded, define syllables by selecting their component <phone>'s and, if stressed, by assigning the proper stress mark
  - Automatic segmentation:
    - listen to the speech sound and transcribe it as a sequence of phones
    - apply the phonetic aligner to the speech signal with its phonetic transcription and obtain its phonetic segmentation
    - import the phonetic segmentation in the MATE environment
    - define syllables as in step 6 above.
- creation notes:

- Authors: Silvia Quazza, Juan María Garrido
- Version: 1., October 1999
- Comments: none
- Literature:

2. Phonetic Representation of Intonation - F0 Coding Module

- name: F0 Coding
- coding purpose: coding of raw f0 values.
- coding level: Prosody
- data sources: spoken corpora (speech files, f0 files obtained by pitch tracking tools)
- module references: none
- description: Here we define the element `<f0>` to represent raw f0 values, whose sequence provides the so-called f0 contour of the utterance.
- example:

f0.xml

```
<f0 id=" f0 _001" value="207" start="130" end="130"/>
<f0 id=" f0_002" value="210" start="140" end="140"/>
<f0 id=" f0_003" value="208" start="150" end="150"/>
<f0 id=" f0_004" value="208" start="160" end="160"/>
```

- markup declaration:

```
ELEMENT <f0>

ATTRIBUTES:
id [ASCII]
value [FLOAT]
start [FLOAT]
end [FLOAT]
```

- coding procedure: import the f0 values from an external file, obtained by a pitch extraction tool
- creation notes:
  - Authors: Silvia Quazza, Juan María Garrido
  - Version: 1., October 1999
  - Comments: none
  - Literature:

3. Phonetic Representation of Intonation - IPO coding module

- name: IPO Annotation
- coding purpose: symbolic annotation of stylized f0 contours as a sequence of pitch movements, according to the the IPO scheme
- coding level: Prosody
- data sources: spoken corpora (speech files, f0 stylised contour files, phonetic transcription)
- module references: f0 coding module (optional), phonetic transcription module
- description: Here we consider two hierarchically ordered elements:
    - `<closecopy>`, representing the inflection points in the stylized curve
    - `<pitmove>`, representing the classified movements from one inflection point to the next one.

  In principle, `<closecopy>` could be directly imported from f0 stylised files or linked to one `<f0>` element, and `<pitmove>` should be linked to two consecutive `<closecopy>` elements.
- example: The following example shows the coding of the Italian sentence *"quell'artificio contabile sara' scoperto facilmente*" read by a female speaker.

closecopy.xml

```
<closecopy id="clscpy_001" value="207" start="130" end="130"/>
<closecopy id="clscpy_002" value="243" start="540" end="540"/>
<closecopy id="clscpy_003" value="285" start="690" end="690"/>
<closecopy id="clscpy_004" value="212" start="860" end="860"/>
<closecopy id="clscpy_005" value="189" start="1110" end="1110"/>
<closecopy id="clscpy_006" value="159" start="1290" end="1290"/>
<closecopy id="clscpy_007" value="209" start="1500" end="1500"/>
<closecopy id="clscpy_008" value="206" start="1750" end="1750"/>
<closecopy id="clscpy_009" value="246" start="2070" end="2070"/>
<closecopy id="clscpy_010" value="226" start="2600" end="2600"/>
<closecopy id="clscpy_011" value="148" start="2780" end="2780"/>
<closecopy id="clscpy_012" value="144" start="3070" end="3070"/>
```

pitmove.xml

```
<pitmove id="pitm_001" type="4" href="closecopy.xml# id(clscpy_001).. id(clscpy_002)"/>
<pitmove id="pitm_001" type="1" href="closecopy.xml# id(clscpy_002).. id(clscpy_003)"/>
<pitmove id="pitm_001" type="B" href="closecopy.xml# id(clscpy_003).. id(clscpy_004)"/>
<pitmove id="pitm_001" type="Ø" href="closecopy.xml# id(clscpy_004).. id(clscpy_005)"/>
<pitmove id="pitm_001" type="B" href="closecopy.xml# id(clscpy_005).. id(clscpy_006)"/>
<pitmove id="pitm_001" type="4" href="closecopy.xml# id(clscpy_006).. id(clscpy_007)"/>
<pitmove id="pitm_001" type="Ø" href="closecopy.xml# id(clscpy_007).. id(clscpy_008)"/>
<pitmove id="pitm_001" type="4" href="closecopy.xml# id(clscpy_008).. id(clscpy_009)"/>
<pitmove id="pitm_001" type="0" href="closecopy.xml# id(clscpy_009).. id(clscpy_010)"/>
<pitmove id="pitm_001" type="B" href="closecopy.xml# id(clscpy_010).. id(clscpy_011)"/>
<pitmove id="pitm_001" type="Ø" href="closecopy.xml# id(clscpy_011).. id(clscpy_012)"/>
```

- markup declaration:

```
ELEMENT <closecopy>

ATTRIBUTES:
id [ASCII]
value [FLOAT]
href <f0> ***< (optional) >***
start [FLOAT]
end [FLOAT]


ELEMENT <pitmove>

ATTRIBUTES:
id [ASCII]
type 0, Ø, 1, 2, 3, 4, 5, A, B, C, D, E, &2, &3, &4, &A, &C, &D
href <closecopy>..<closecopy>
start [FLOAT]
end [FLOAT]
```

- coding procedure: The most IPO-conformant coding procedure will directly import the stylized f0 curve, obtained with the help of a proper external environment for perceptual, using the `<closecopy>` element with no need of the `<f0>` element, and will consist in the following steps:
    - open the speech file in order to listen to its intonation
    - open the corresponding phonetic segmentation (`<phone>` and `<syllable>`)
    - import the close copy and display it as a curve, aligned with phonetic segmentation
    - define `<pitmove>` elements by selecting the segments of the stylized curve (delimited by two consecutive <closecopy> elements) and labeling each of them according to the following criteria:
        - if it can be considered to coincide with the ideal baseline or topline, by a global look at the curve, label it 0 or Ø respectively
        - otherwise choose the proper label on the basis of movement direction and size and of its position in the syllable, judged by looking at its phonetic alignment

    If the close copy is not available, the third step may be replaced by the following steps (a very simplified approximation of the correct stylization procedure):
    - import or generate automatically the raw f0 curve and display it
    - obtain a closecopy by selecting the 'relevant' `<f0>` points on the raw curve; base such stylization on the shape of the curve, the perceived intonation of the sound file and the alignment with syllables (accents, boundaries...)
- creation notes:

- Authors: Silvia Quazza, Juan María Garrido
- Version: 1., October 1999
- Comments: none
- Literature:

4. Phonetic Representation of Intonation - INTSINT Coding Module

- name: INTSINT Annotation
- coding purpose: symbolic coding of stylized f0 contours, where each target point is coded as an absolute or relative tone, according to the INTSINT scheme
- coding level: Prosody
- data sources: spoken corpora (speech files, f0 stylised contour files, phonetic transcription)
- module references: f0 coding module, phonetic transcription module (both optional)
- description: The elements necessary to represent the INTSINT notation system are the following:
  - `<momel>`, for the inflection points in the stylized curve
  - `<intone>`, for the labeled tones

  There is a one-to-one mapping between `<intone>`'s and `<momel>`'s. The alignment with the soundfile may be kept through the base element `<f0>` or, in case the <momel> stylized curve is directly imported, the link with `<f0>` can be skipped and `<momel>` can be directly aligned with the soundfile.
- example: The example presented here shows the MOMEL and INTSINT annotation of the French utterance '*Il faut que je sois a Grenoble Samedi vers quinze heures*', using the `<momel>` and `<intone>` elements.

momel.xml

```
<momel id="mml_001" value="163" start="106" end="106"/>
<momel id="mml_002" value="217" start="265" end="265"/>
<momel id="mml_003" value="148" start="521" end="521"/>
<momel id="mml_004" value="190" start="617" end="617"/>
<momel id="mml_005" value="130" start="827" end="827"/>
<momel id="mml_006" value="223" start="1249" end="1249"/>
<momel id="mml_007" value="139" start="1614" end="1614"/>
<momel id="mml_008" value="172" start="1822" end="1822"/>
<momel id="mml_009" value="144" start="1983" end="1983"/>
<momel id="mml_010" value="185" start="2078" end="2078"/>
<momel id="mml_011" value="152" start="2248" end="2248"/>
<momel id="mml_012" value="99" start="2505" end="2505"/>
<momel id="mml_013" value="152" start="2730" end="2730"/>
```

intone.xml

```
<intone id="intn_001" type="L" href="momel.xml# id(mml_001)"/>
<intone id="intn_002" type="T" href="momel.xml# id(mml_002)"/>
<intone id="intn_003" type="M" href="momel.xml# id(mml_003)"/>
<intone id="intn_004" type="H" href="momel.xml# id(mml_004)"/>
<intone id="intn_005" type="L" href="momel.xml# id(mml_005)"/>
<intone id="intn_006" type="T" href="momel.xml# id(mml_006)"/>
<intone id="intn_007" type="M" href="momel.xml# id(mml_007)"/>
<intone id="intn_008" type="H" href="momel.xml# id(mml_008)"/>
<intone id="intn_009" type="L" href="momel.xml# id(mml_009)"/>
<intone id="intn_010" type="H" href="momel.xml# id(mml_010)"/>
<intone id="intn_011" type="D" href="momel.xml# id(mml_011)"/>
<intone id="intn_012" type="B" href="momel.xml# id(mml_012)"/>
<intone id="intn_013" type="M" href="momel.xml# id(mml_013)"/>
```

- markup declaration:

```
ELEMENT <momel>

ATTRIBUTES:
id [ASCII]
value [FLOAT]
href <f0> (optional)
start [FLOAT]
end [FLOAT]

ELEMENT <intone>

ATTRIBUTES:
id [ASCII]
type T, M, B, H, S, L, U, D
href <momel>
start [FLOAT]
end [FLOAT]
```

- coding procedure: A specific tool, 'mes', has been developed to perform automatic intonation transcription according to the INTSINT system. Both stylization and annotation can be performed automatically by 'mes'. So, the simplest way to get to INTSINT annotation in the MATE environment would be the following:
    - import the MOMEL stylized curve in `<momel>`
    - import the INTSINT annotation in `<intone>`
    - link `<momel>` to `<f0>` and `<intone>` to `<momel>` (an automatic function should be provided for that by the workbench)

  In case only `<momel>` is imported, `<intone>`'s may be created manually by the following procedure:
    - open the speech file in order to listen to its intonation
    - open the corresponding phonetic segmentation
    - import `<momel>` elements and display them as a stylized curve

    - define <intone> elements by selecting every <momel> element (inflection point in the stylized curve) and mark it with the proper label

- creation notes:
    - Authors: Silvia Quazza, Juan María Garrido
    - Version: 1., October 1999
    - Comments: none
    - Literature:

4. Phonetic Representation of Intonation - INTSINT Coding Module

- name: INTSINT Annotation
- coding purpose: symbolic coding of stylized f0 contours, where each target point is coded as an absolute or relative tone, according to the INTSINT scheme
- coding level: Prosody
- data sources: spoken corpora (speech files, f0 stylised contour files, phonetic transcription)
- module references: f0 coding module, phonetic transcription module (both optional)
- description: The elements necessary to represent the INTSINT notation system are the following:
    - <momel>, for the inflection points in the stylized curve
    - <intone>, for the labeled tones

  There is a one-to-one mapping between <intone>'s and <momel>'s. The alignment with the soundfile may be kept through the base element <f0> or, in case the <momel> stylized curve is directly imported, the link with <f0> can be skipped and <momel> can be directly aligned with the soundfile.
- example: The example presented here shows the MOMEL and INTSINT annotation of the French utterance '*Il faut que je sois a Grenoble Samedi vers quinze heures*', using the <momel> and <intone> elements.

momel.xml

```
<momel id="mml_001" value="163" start="106" end="106"/>
<momel id="mml_002" value="217" start="265" end="265"/>
<momel id="mml_003" value="148" start="521" end="521"/>
<momel id="mml_004" value="190" start="617" end="617"/>
<momel id="mml_005" value="130" start="827" end="827"/>
<momel id="mml_006" value="223" start="1249" end="1249"/>
<momel id="mml_007" value="139" start="1614" end="1614"/>
<momel id="mml_008" value="172" start="1822" end="1822"/>
```

```
<momel id="mml_009" value="144" start="1983" end="1983"/>
<momel id="mml_010" value="185" start="2078" end="2078"/>
<momel id="mml_011" value="152" start="2248" end="2248"/>
<momel id="mml_012" value="99" start="2505" end="2505"/>
<momel id="mml_013" value="152" start="2730" end="2730"/>
```

intone.xml

```
<intone id="intn_001" type="L" href="momel.xml# id(mml_001)"/>
<intone id="intn_002" type="T" href="momel.xml# id(mml_002)"/>
<intone id="intn_003" type="M" href="momel.xml# id(mml_003)"/>
<intone id="intn_004" type="H" href="momel.xml# id(mml_004)"/>
<intone id="intn_005" type="L" href="momel.xml# id(mml_005)"/>
<intone id="intn_006" type="T" href="momel.xml# id(mml_006)"/>
<intone id="intn_007" type="M" href="momel.xml# id(mml_007)"/>
<intone id="intn_008" type="H" href="momel.xml# id(mml_008)"/>
<intone id="intn_009" type="L" href="momel.xml# id(mml_009)"/>
<intone id="intn_010" type="H" href="momel.xml# id(mml_010)"/>
<intone id="intn_011" type="D" href="momel.xml# id(mml_011)"/>
<intone id="intn_012" type="B" href="momel.xml# id(mml_012)"/>
<intone id="intn_013" type="M" href="momel.xml# id(mml_013)"/>
```

- markup declaration:

```
ELEMENT <momel>

ATTRIBUTES:
id [ASCII]
value [FLOAT]
href <f0> (optional)
start [FLOAT]
end [FLOAT]
```

```
ELEMENT <intone>

ATTRIBUTES:
id [ASCII]
type T, M, B, H, S, L, U, D
href <momel>
start [FLOAT]
end [FLOAT]
```

- coding procedure: A specific tool, 'mes', has been developed to perform automatic intonation transcription according to the INTSINT system. Both stylization and annotation can be performed automatically by 'mes'. So, the simplest way to get to INTSINT annotation in the MATE environment would be the following:
    - import the MOMEL stylized curve in `<momel>`
    - import the INTSINT annotation in `<intone>`
    - link `<momel>` to `<f0>` and `<intone>` to `<momel>` (an automatic function should be provided for that by the workbench)

In case only <momel> is imported, <intone>'s may be created manually by the following procedure:

- open the speech file in order to listen to its intonation
- open the corresponding phonetic segmentation
- import <momel> elements and display them as a stylized curve
- define <intone> elements by selecting every <momel> element (inflection point in the stylized curve) and mark it with the proper label

- creation notes:
  - Authors: Silvia Quazza, Juan María Garrido
  - Version: 1., October 1999
  - Comments: none
  - Literature:

5. Phonological Representation of Intonation - ToBI (Tone Layer) Coding Module

- name: ToBI (Tone Layer) Annotation
- coding purpose: phonological annotation of the intonation curve, distinguishing pitch accents, phrase accents and boundary tones, according to the ToBI scheme
- coding level: Prosody
- data sources: spoken corpora (speech files, f0 files, orthographic transcription, phonetic transcription)
- module references: orthographic transcription module, phonetic transcription module, f0 coding module
- description: In our XML adaptation of ToBI, four elements have been defined:
  - <tobitone>, for the *tones*, distinguished according to their function as *pitch accents*, *phrase accents* or *boundary tones* and labeled according to a classification of their linguistically admissible types
  - <target>, to mark peak location when it occurs outside the scope of the accented syllable
  - <f0range>, to mark the highest f0 value in the curve
  - <repair>, to mark the restart of the intonation contour after a disfluency

The four elements are not hierarchically ordered. All *may* refer to the f0 curve, while

only the two accessory element <target> and <f0range> are *necessarily* linked to <f0>. The <tobitone> and <repair> elements can be linked to prosodic units and/or to phonetic descriptions of intonation, rather than raw f0.

- example: The following example shows the ToBI annotation of the English utterance "Show me the cheapest fare from Philadelphia to Dallas excluding restriction VU slash one" (obtained from the TOBI-TRAINING material), using the elements <tobitone> and <repair>.

tobitone.xml

```
<tobitone id="tbtn_001" type="H*"   class="pitaccent" href="word.xml#id(wrd_001)"
                                                      start="2052" end="2052"/>
<tobitone id="tbtn_002" type="L+H*" class="pitaccent" href="word.xml#id(wrd_004)"
                                                      start="2579" end="2579"/>
<tobitone id="tbtn_003" type="!H*"  class="pitaccent" href="word.xml#id(wrd_005)"
                                                      start="3065" end="3065"/>
<tobitone id="tbtn_004" type="L-"   class="phraccent" href="word.xml#id(wrd_005)"
                                                      start="3315" end="3315"/>
<tobitone id="tbtn_005" type="L%"   class="boundtone" href="word.xml#id(wrd_005)"
                                                      start="3315" end="3315"/>
<tobitone id="tbtn_006" type="L+H*" class="pitaccent" href="word.xml#id(wrd_009)"
                                                      start="4470" end="4470"/>
<tobitone id="tbtn_007" type="!H*"  class="pitaccent" href="word.xml#id(wrd_009)"
                                                      start="4771" end="4771"/>
<tobitone id="tbtn_008" type="L-"   class="phraccent" href="word.xml#id(wrd_009)"
                                                      start="5015" end="5015"/>
<tobitone id="tbtn_009" type="H*"   class="pitaccent" href="word.xml#id(wrd_011)"
                                                      start="5388" end="5388"/>
<tobitone id="tbtn_010" type="L-"   class="phraccent" href="word.xml#id(wrd_011)"
                                                      start="5855" end="5855"/>
<tobitone id="tbtn_011" type="L%"   class="boundtone" href="word.xml#id(wrd_011)"
                                                      start="5855" end="5855"/>
<tobitone id="tbtn_012" type="L+H*" class="pitaccent" href="word.xml#id(wrd_012)"
                                                      start="6984" end="6984"/>
<tobitone id="tbtn_013" type="L-"   class="phraccent" href="word.xml#id(wrd_012)"
                                                      start="7399" end="7399"/>
<tobitone id="tbtn_014" type="L%"   class="boundtone" href="word.xml#id(wrd_012)"
                                                      start="7399" end="7399"/>
<tobitone id="tbtn_015" type="H*"   class="pitaccent" href="word.xml#id(wrd_013)"
                                                      start="8154" start="8154"/>
<tobitone id="tbtn_016" type="L-"   class="phraccent" href="word.xml#id(wrd_013)"
                                                      start="8585" end="8585"/>
<tobitone id="tbtn_017" type="L%"   class="boundtone" href="word.xml#id(wrd_013)"
                                                      start="8585" end="8585"/>
<tobitone id="tbtn_018" type="H*"   class="pitaccent" href="word.xml#id(wrd_014)"
                                                      start="8711" end="8711"/>
<tobitone id="tbtn_019" type="!H*"  class="pitaccent" href="word.xml#id(wrd_015)"
                                                      start="8928" end="8928"/>
<tobitone id="tbtn_020" type="L-"   class="phraccent" href="word.xml#id(wrd_015)"
                                                      start="9114" end="9114"/>
<tobitone id="tbtn_021" type="H*"   class="pitaccent" href="word.xml#id(wrd_016)"
                                                      start="9353" end="9353"/>
<tobitone id="tbtn_022" type="H*"   class="pitaccent" href="word.xml#id(wrd_017)"
                                                      start="9694" end="9694"/>
<tobitone id="tbtn_023" type="L-"   class="phraccent" href="word.xml#id(wrd_017)"
                                                      start="9880" end="9880"/>
<tobitone id="tbtn_024" type="L%"   class="boundtone" href="word.xml#id(wrd_017)"
                                                      start="9880" end="9880"/>
```

repair.xml

```
<repair id="rpr_001" type="%r" start="4149" end="4149"/>
```

- markup declaration:

ELEMENT <tobitone>

ATTRIBUTES:

```
id [ASCII]
type L-, H-, !H-, -, -?, X-?, L-L%, L-H%, H-H%, H-L%, %, %?, X%?, H*,
!H*, L*, L*+H, L*+!H, L+H*, L+!H*, H+!H*, *, *?, X*?
class pitaccent, phraccent, boundtone
href <f0> or <closecopy> or <momel> or <intone> or <syllable> or <word>
start [FLOAT]
end [FLOAT]
```

The set of symbols defined for the attribute ‘type’ includes the allowable combination of pitch accents, phrase accents, boundary tones and/or uncertainty symbols, as defined in the ToBI guidelines. The value for the attribute 'type' should be consistent with the attribute 'class', according to the semantics of the different labels described in the tables in 6.2.1.

```
ELEMENT <target>

ATTRIBUTES:

id [ASCII]
type EarlyF0, LateF0
href <f0> or
<closecopy> or
<momel> or
<intone>
start [FLOAT]
end [FLOAT]


ELEMENT <f0range>

ATTRIBUTES:
id [ASCII]
type HiF0
href <f0> or
<closecopy> or
<momel> or
<intone>
start [FLOAT]
end [FLOAT]


ELEMENT <repair>

ATTRIBUTES:
id [ASCII]
type %r
href <f0> or
<closecopy> or
<momel> or
<intone> or
<syllable>
start [FLOAT]
end [FLOAT]
```

- coding procedure: Different procedures may be followed to obtain a ToBI annotation of intonation. A possible procedure, assuming that the `<f0>` and `<syllable>` elements are available, is:

- open the following synchronized windows: signal window, <f0> (with its graphical display), <word>, <syllable> by listening to the signal and inspecting the f0 curve and the aligned words and syllables, find out pitch accents, phrase boundaries, boundary tones and repairs (according to ToBI definitions and guidelines)
- for each detected intonation event, select the <syllable> on which it occurs, create the corresponding (linked) <tobitone> or <repair> element and assign it the proper class attribute; time values will be inherited from <syllable> (or, if desired, can be set explicitly in correspondence of the f0 peak or valley)
- in case the peak or valley of the event is outside the syllable, choose its exact <f0> point and create the linked <target> element
- for each phrase, find out its f0 maximum, select its <f0> point and create the linked <f0range> element

- creation notes:
  - Authors: Silvia Quazza, Juan María Garrido
  - Version: 1., October 1999
  - Comments: none
  - Literature:

6. Prosodic Phrasing - ToBI (Break Index) Coding Module

- name: ToBI (Break Index) Annotation
- coding purpose: mark the boundaries between intonation phrases and rate their depth, according to the ToBI scheme
- coding level: Prosody
- data sources: spoken corpora (speech files, orthographic transcription, f0 files)
- module references: orthographic transcription coding module, f0 coding module
- description: A single element <breakindex> is here defined to represent word boundaries and rate them with the proper degree of disjuncture.
- example: The following example shows the 'break index' annotation of the same utterance of section 6.5 ("*Show me the cheapest fare from Philadelphia to Dallas excluding restriction VU slash one*") using the <breakindex> element:

breakindex.xml

```
<breakindex id="brkndx_001" type="1" href=word.xml# id(wrd_001) start="2105" end="2105"/>
<breakindex id="brkndx_002" type="1" href=word.xml# id(wrd_002) start="2245" end="2245"/>
<breakindex id="brkndx_003" type="1" href=word.xml# id(wrd_003) start="2355" end="2355"/>
```

```
<breakindex id="brkndx_004" type="1" href=word.xml# id(wrd_004) start="2935" end="2935"/>
<breakindex id="brkndx_005" type="4" href=word.xml# id(wrd_005) start="3315" end="3315"/>
<breakindex id="brkndx_006" type="1" href=word.xml# id(wrd_006) start="3565" end="3565"/>
<breakindex id="brkndx_007" type="1p" href=word.xml# id(wrd_007) start="3836" end="3836"/>
<breakindex id="brkndx_008" type="1" href=word.xml# id(wrd_008) start="4325" end="4325"/>
<breakindex id="brkndx_009" type="3" href=word.xml# id(wrd_009) start="5015" end="5015"/>
<breakindex id="brkndx_010" type="1" href=word.xml# id(wrd_010) start="5225" end="5225"/>
<breakindex id="brkndx_011" type="4" href=word.xml# id(wrd_011) start="5855" end="5855"/>
<breakindex id="brkndx_012" type="4" href=word.xml# id(wrd_012) start="7399" end="7399"/>
<breakindex id="brkndx_013" type="4" href=word.xml# id(wrd_013) start="8585" end="8585"/>
<breakindex id="brkndx_014" type="1" href=word.xml# id(wrd_014) start="8825" end="8825"/>
<breakindex id="brkndx_015" type="3" href=word.xml# id(wrd_015) start="9115" end="9115"/>
<breakindex id="brkndx_016" type="1" href=word.xml# id(wrd_016) start="9595" end="9595"/>
<breakindex id="brkndx_017" type="4" href=word.xml# id(wrd_017) start="9880" end="9880"/>
```

- markup declaration:

```
ELEMENT <breakindex>

ATTRIBUTES:
id [ASCII]
type 0, 1-, 1, 1p, 1p?, 2-, 2, 2p, 2p?, 3-, 3, 3p, 3p?, 4-, 4, X
href <word>
start [FLOAT]
end [FLOAT]
```

The attribute ‘type’ can only contain an allowable combination of ToBI symbols, as described in the ToBI guidelines.

- coding procedure: A procedure for break index annotation may be:
    - open the following synchronized windows: speech file, f0 curve (`<f0>` or `<closecopy>` or `<momel>`...), `<word>`
    - select a word to define a corresponding `<breakindex>` element
    - listen to a surrounding portion of the speech signal and look at the f0 curve, in order to classify the boundary depth and choose the proper label
- creation notes:
    - Authors: Silvia Quazza, Juan María Garrido
    - Version: 1., October 1999
    - Comments: none

      Literature:

Morposyntax

Edited Transcription Coding Module

name: Edited Transcription (ET)

coding purpose: to code disfluency phenomena in speech

coding level: Morphosyntax

data sources: spoken corpora

module references: orthographic transcription module

description: four elements are used to annotate disfluency phenomena. seg elements are all-purpose elements intended to mark dysfluent portions of dialogue and their possible repairs (when present in context). Attribute type identifies the specific type of disfluency which is found in the annotated segment, namely whether it is an interruption, a non-standard use, an omission, or a completion of a previous utterance. Attribute rep allows the annotator to indicate the target or standard form of a non-standard usage. Attribute ins allows to insert missing elements. seg elements convey basic, obligatory information. Further refinements of this obligatory information are possible through use of recommended and optional elements, which refer to seg elements through inline href links, namely dys, reparandum, signal and repair. dys elements serve the purpose of specifying the type of relationship between two seg elements, when these are used to mark dysfluencies which are contiguous in nature. An attribute type can be used to further define the type of disfluency. The elements reparandum, signal and repair are to be seen as a means for a more detailed analysis of the components of a dysfluency. By referring to and qualifying seg elements, they serve the purpose of specifying which previously identified seg element is repaired, which element is signalling that a repairing sequence is about to be uttered, and which element corresponds to the repair in the strict sense.

example:

given this input…:

```
<w id="w_001"> I </w>
<w id="w_002"> wanted </w>
<w id="w_003"> uh </w>
<w id="w_004"> I </w>
<w id="w_005"> thought </w>
<w id="w_006"> I </w>
<w id="w_007"> wanted </w>
<w id="w_008"> to </w>
<w id="w_009"> invite </w>
<w id="w_010"> Margie </w>
```

…the following annotation is built:

```
<seg id="seg_001" type="broken" href="orth.xml#id(w_001)..id(w_002)"/>
<seg id="seg_002" href="orth.xml#id(w_004)..id(w_010)"/>
<dys id="dys_001" type="retrins" href="edit.xml#id(seg_001)..(seg_002)">
    <reparandum id="repm_001" href="edit.xml#id(seg_001)"/>
    <repair id="rep_001" href="edit.xml#id(seg_002)"/>
</dys>
```

markup declaration:

```
ELEMENT edit_file (seg+, dys+)
```

```
ELEMENT seg
ATTRIBUTES:
type (broken | sic | gap | scomp | ocomp)
rep TEXT
ins TEXT
ID
HREF
```

Recommended extensions to the core scheme:

```
ELEMENT dys (repair?, reparandum?, signal?)
ATTRIBUTES:
type TEXT
ID
HREF
```

Optional extensions:

```
ELEMENT reparandum
ATTRIBUTES:
ID
HREF

ELEMENT signal
ATTRIBUTES:
ID
HREF
ELEMENT repair
ATTRIBUTES:
ID
HREF
```

coding procedure: Encode by coder 1. Check by coder 2.

creation notes:

Authors: Claudia Soria, Vito Pirrelli
Version: 1., May 1999; 2., October 1999
Comments: none
Literature:

Morphosyntactic Annotation Coding Module

name: Morphosyntactic Annotation

coding purpose: identification of morphological words, annotation of part-of-speech categories, annotation of morpho-syntactic features, annotation of interrupted words, annotation of clitics, annotation of compound words, annotation of derivational morphology.

coding level: Morphosyntax

data sources: spoken or written corpora

module references: orthographic transcription module

description: six elements are used to annotate morphological analysis . mw elements identify morphological words. Attribute type is mandatory: it specifies the part-of-speech category of an item. In this implementation, type is used to encode EAGLES-conformant part-of-speech categories; attribute subtype is optional, and may be used to specify additional

morphosyntactic features to be associated with words. In the actual implementation presented here, subtype is used to convey EAGLES-conformant recommended morpho-syntactic values. Finally, attribute lemma allows for specification of the lemma of the item in question. An optional attribute broken serves to annotate word partials.cpw elements are used to annotate compounds. Attributes are the same as those for mw elements. A cpw_h element is used to mark the semantic head in a compound. Three elements, namely stem, prefix and suffix are used to annotate derivational morphology.

example:

```
<mw id="mw_001" type="PD" subtype="PD1020115" lemma="we" href="orth.xml#id(w_001)"/>
<mw id="mw_002" type="V" subtype="V00011111" lemma="take" href="orth.xml#id(w_002)"/>
<mw id="mw_003" type="AT" subtype="AT1000" lemma="the" href="orth.xml#id(w_003)"/>
<mw id="mw_004" type="N" subtype="N102000" lemma="orange" href="orth.xml#id(w_004)"/>
<mw id="mw_005" type="AP" subtype="AP1" lemma="to" href="orth.xml#id(w_005)"/>
<mw id="mw_006" type="N" subtype="N201000" lemma="Elmira" href="orth.xml#id(w_006)"/>
<mw id="mw_007" type="I" subtype="I" href="orth.xml#id(w_007)"/>
<mw id="mw_008" type="PD" subtype="PD1010115" lemma="I" href="orth.xml#id(w_008)"/>
<mw id="mw_009" type="V" subtype="V00011111" lemma="mean" href="orth.xml#id(w_009)"/>
<mw id="mw_010" type="AP" subtype="AP1" lemma="to" href="orth.xml#id(w_010)"/>
<mw id="mw_011" type="N" subtype="N201000" lemma="Corning" href="orth.xml#id(w_011)"/>
<mw id="mw_001" type="PD" subtype="PD1020115" lemma="we" href="orth.xml#id(w_001)"/>
<mw id="mw_002" type="V" subtype="V00011111" lemma="take" href="orth.xml#id(w_002)"/>
<mw id="mw_003" type="AT" subtype="AT1000" lemma="the" href="orth.xml#id(w_003)"/>
<mw id="mw_004" type="N" subtype="N102000" lemma="orange" href="orth.xml#id(w_004)"/>
<mw id="mw_005" type="AP" subtype="AP1" lemma="to" href="orth.xml#id(w_005)"/>
<mw id="mw_006" type="N" subtype="N201000" lemma="Elmira" href="orth.xml#id(w_006)"/>
<mw id="mw_007" type="I" subtype="I" href="orth.xml#id(w_007)"/>
<mw id="mw_008" type="PD" subtype="PD1010115" lemma="I" href="orth.xml#id(w_008)"/>
<mw id="mw_009" type="V" subtype="V00011111" lemma="mean" href="orth.xml#id(w_009)"/>
<mw id="mw_010" type="AP" subtype="AP1" lemma="to" href="orth.xml#id(w_010)"/>
<mw id="mw_011" type="N" subtype="N201000" lemma="Corning" href="orth.xml#id(w_011)"/>
```

markup declaration:

```
ELEMENT mw (lexit*, stem*, suffix*, prefix*)

ATTRIBUTES
type (N|V|AJ|PD|AT|AV|AP|C|NU|I|U|R|F|DM|PU)
lemma TEXT
subtype TEXT
broken (Y|N)
ID
HREF


ELEMENT cpw (cpw_h?)

ATTRIBUTES
type (N|V|AJ|PD|AT|AV|AP|C|NU|I|U|R|F|DM|PU)
subtype TEXT
roken (Y|N)
ID
HREF


ELEMENT cpw_h

ATTRIBUTES
ID
HREF


ELEMENT stem
```

```
ATTRIBUTES

type (N|V)
ID
HREF


ELEMENT suffix

ATTRIBUTES

ID
HREF


ELEMENT prefix

ATTRIBUTES

ID
HREF
```

The following element is used in case there is a reference lexicon in xml format

ELEMENT lexit

ATTRIBUTES
ID
HREF

coding procedure: morphological annotation is almost always performed automatically. Manual checking is recommended.

creation notes:

Authors: Claudia Soria, Vito Pirrelli
Version: 1., May 1999; 2., October 1999
Comments: none

Chunking Coding Module

name: Chunking

coding purpose: to code syntactic structure in terms of labelled entities corresponding to chunks. Each chunk is further analyzed for its internal structure.

coding level: Morphosyntax

data sources: spoken or written corpora

module references: morphosyntactic annotation module

description: seven elements are used to annotate syntactic analysis. ch elements are used to identify a sequence of adjacent word tokens which are mutually related through dependency links (i.e., a chunk). Two attributes are used for the description of chunks: type is mandatory, and encodes the syntactic category to which a given chunk belongs. broken is optional, and serves to annotate chunk partials. potgov elements identify "potential governors", namely the lexical heads of chunks. aux, cop, intro, modal and causal elements specify, respectively, the auxiliary verb, the copula, the introducer or preposition, the modal auxiliary verb and the causative verb in a chunk, if applicable.

example:

given this input…:

```
<mw id="mw_001"> hello </mw>
<mw id="mw_002"> can </mw>
<mw id="mw_003"> I </mw>
<mw id="mw_004"> help </mw>
<mw id="mw_005"> you </mw>
```

…the following annotation is built:

```
<ch id="ch_001" type="ADV" href="mword.xml#id(mw_001)">
  <potgov id=”p_001” href=” mword.xml#id(mw_001)”/>
</ch>
<ch id="ch_002" type="FV" href="mword.xml#id(mw_002)">
  <potgov id=”p_002” href=” mword.xml#id(mw_002)”/>
</ch>
<ch id="ch_003" type="N" href="mword.xml#id(mw_003)">
  <potgov id=”p_003” href=” mword.xml#id(mw_003)”/>
</ch>
<ch id="ch_004" type="FV" href="mword.xml#id(mw_004)">
  <potgov id=”p_004” href=” mword.xml#id(mw_004)”/>
</ch>
<ch id="ch_005" type="N" href="mword.xml#id(mw_005)">
  <potgov id=”p_005” href=” mword.xml#id(mw_005)”/>
</ch>
```

markup declaration:

```
ELEMENT ch (potgov, aux?, cop?, intro?, modal?, caus?)

ATTRIBUTES
type (ADJ|PA|ADV|SUBORD|N|P|FV|G|I|PART|Di|ADJ_PART|COORD|U)
broken (Y | N)
ID
HREF


ELEMENT potgov

ATTRIBUTES
ID
HREF
ELEMENT aux
ID
HREF


ELEMENT cop
```

```
ATTRIBUTES
ID
HREF


ELEMENT intro

ATTRIBUTES
ID
HREF


ELEMENT modal

ATTRIBUTES
ID
HREF


ELEMENT caus

ATTRIBUTES
ID
HREF
```

coding procedure: Chunking can be performed either automatically or manually. In the first case, a manual checking of the chunker output is recommended. In the second case, the standard practice is sufficient (i.e., encode by coder 1. check by coder 2.)

creation notes:

Authors: Claudia Soria, Vito Pirrelli
Version: 1., May 1999; 2., October 1999
Comments: none
Literature:

Functional Annotation Coding Module

name: Functional Annotation

coding purpose: to encode functional analysis of data, that is to provide information about how grammatical relations such as subject, object and indirect object are instantiated in context.

coding level: Morphosyntax

data sources: spoken or written corpora

module references: morphosyntactic annotation module

description: Encoding is carried out by means of funct elements, which point to lexical tokens only indirectly.

The terms of the relationship are annotated through two dedicated elements, head and dep, which are hierarchically embedded within funct elements and point to the relevant lexical tokens in the resource file.
The type of relationship involved is represented by means of a list of values for the attribute type, further specifying dep elements.
Morphosyntactic features can be specified, when needed, through attributes in head and dep elements.
head attributes are: diath (i.e., the diathesis of a verbal head, whether active, passive, or middle), tense, person, number and gender, respectively the morphosyntactic tense, person, number and gender of the head.
dep attributes are: intro (for introducer, i.e. the element which possibly introduces the dependent), case (i.e., the case of the dependent), and
synt_real (i.e., the particular syntactic realization of the dependent, whether clausal or non clausal).

example:

given this input…:

```
<mw id="mw_001"> Paul </mw>
<mw id="mw_002"> said </mw>
<mw id="mw_003"> that </mw>
<mw id="mw_004"> he </mw>
<mw id="mw_005"> will </mw>
<mw id="mw_006"> accept </mw>
<mw id="mw_007"> the </mw>
<mw id="mw_008"> job </mw>
```

..we build the following annotation:

```
<funct id="funct_001" >
<head id="h_001" href="mword.xml#id(mw_002)"/>
<dep id="d_001" type="subj" href="mword.xml#id(mw_001)"/>
<dep id="d_002" type="comp" href="mword.xml#id(mw_006)"/>
</funct>
<funct id="funct_002">
<head id="h_002" href="mword.xml#id(mw_006)"/>
<dep id="d_003" type="subj" href="mword.xml#id(mw_004)"/>
<dep id="d_004" type="dobj" href="mword.xml#id(mw_008)"/>
</funct>
```

markup declaration:

```
ELEMENT funct (head, dep+)

ATTRIBUTES:
ID
HREF


ELEMENT head

ATTRIBUTES:
head TEXT
diath (active|passive|middle)
person (1|2|3)
number (sg|pl)
gender (m|f|n)
v_type (impers)
ID
HREF


ELEMENT dep
```

```
ATTRIBUTES:
type (subj|dobj|obj2|iobj|mod|comp)
intro TEXT
case TEXT
synt_real (n_cl|cl|c|x)
ID
HREF


ELEMENT coord (arg+)

ATTRIBUTES:
type (and|or|comma)
ID
HREF


ELEMENT arg

ATTRIBUTES:
ID
HREF


ELEMENT bind (arg+)

ATTRIBUTES:
ID
HREF
```

coding procedure: manual annotation of the functional syntactic analysis of a text is performed through the following steps:

- identify a <head> element
- identify <dep> elements associated with that <head>
- for each <dep> element, specify the dependency type
- enclose the <head> and all its <dep>endents in a single element <funct>
- repeat 1 to 4
- check by coder 2

creation notes:

Authors: Claudia Soria, Vito Pirrelli
Version: 1., May 1999; 2., October 1999
Comments: none
Literature:

Dialogue Acts

1. **Module name:** DAMSL/SWBD-DAMSL-variant dialogue acts [DAMSL-SWBD_DAMSLv-DA].

2. **Module purpose:** Dialogue act coding for two-agent, task-oriented, problem-solving dialogues. Suitable for mass data annotation but still theoretical distinctive.

3. **Coding level:** Dialogue acts.

4. **Data source:** Spoken dialogue corpora.

5. **Module references:** word level/orthographic transcription module.

6. **Markup declaration:**

   Internal scheme (DAMSL):
   ELEMENT segment
   ATTRIBUTES id: ID href: HREF(word level)

   ELEMENT communicative_status (child of segment)
   ATTRIBUTES id:ID

   ELEMENT information_level (child of segment)
   ATTRIBUTES id:ID

   ELEMENT forward_looking_function (child of segment)
   ATTRIBUTES id:ID

   ELEMENT backward_looking_function (child of segment)
   ATTRIBUTES id:ID

   ELEMENT uninterpretable (child of communicative_status)
   ATTRIBUTES id:ID

   ELEMENT abandoned (child of communicative_status)
   ATTRIBUTES id:ID

   ELEMENT self_talk (child of communicative_status)
   ATTRIBUTES id:ID

   ELEMENT task (child of information_level)
   ATTRIBUTES id:ID

   ELEMENT task_management (child of information_level)

ATTRIBUTES id:ID

ELEMENT communication_management (child of information_level)
ATTRIBUTES id:ID

ELEMENT other_level (child of information_level)
ATTRIBUTES id:ID

ELEMENT statement (child of forward_looking_function)
ATTRIBUTES id:ID

ELEMENT assert (child of statement)
ATTRIBUTES id:ID

ELEMENT reassert (child of statement)
ATTRIBUTES id:ID

ELEMENT other_statement (child of statement)
ATTRIBUTES id:ID

ELEMENT influencing_addressee_future_action
(child of forward_looking_function)
ATTRIBUTES id:ID

ELEMENT open_option (child of influencing_addressee_future_action)
ATTRIBUTES id:ID

ELEMENT action_directive (child of influencing_addressee_future_action)
ATTRIBUTES id:ID

ELEMENT info_request (child of forward_looking_function)
ATTRIBUTES id:ID

ELEMENT committing_speaker_future_action (child of forward_looking_function)
ATTRIBUTES id:ID

ELEMENT offer (child of committing_speaker_future_action)
ATTRIBUTES id:ID

ELEMENT commit (child of committing_speaker_future_action)
ATTRIBUTES id:ID

ELEMENT conventional (child of forward_looking_function)
ATTRIBUTES id:ID

ELEMENT opening (child of conventional)
ATTRIBUTES id:ID

ELEMENT closing (child of conventional)
ATTRIBUTES id:ID

ELEMENT explicite_performative (child of forward_looking_function)
ATTRIBUTES id:ID

ELEMENT exclamation (child of forward_looking_function)
ATTRIBUTES id:ID

ELEMENT other_forward_function (child of forward_looking_function)
ATTRIBUTES id:ID

ELEMENT agreement (child of backward_looking_function)
ATTRIBUTES id:ID

ELEMENT accept (child of agreement)
ATTRIBUTES id:ID

ELEMENT accept_part (child of agreement)
ATTRIBUTES id:ID

ELEMENT maybe (child of agreement)
ATTRIBUTES id:ID

ELEMENT reject_part (child of agreement)
ATTRIBUTES id:ID

ELEMENT reject (child of agreement)
ATTRIBUTES id:ID

ELEMENT hold (child of agreement)
ATTRIBUTES id:ID

ELEMENT understanding (child of backward_looking_function)
ATTRIBUTES id:ID

ELEMENT signal_non_understanding (child of understanding)
ATTRIBUTES id:ID

ELEMENT signal_understanding (child of understanding)
ATTRIBUTES id:ID

ELEMENT acknowledge (child of signal_understanding)
ATTRIBUTES id:ID

ELEMENT repeat_rephrase (child of signal_understanding)
ATTRIBUTES id:ID

ELEMENT completion (child of signal_understanding)
ATTRIBUTES id:ID

ELEMENT correct_misspeaking (child of understanding)
ATTRIBUTES id:ID

ELEMENT answer (child of backward_looking_function)
ATTRIBUTES id:ID

Surface scheme (variant of SWBD-DAMSL):

ELEMENT segment
ATTRIBUTES id:ID href: HREF(word level)

ELEMENT uninterpretable (child of segment)
ATTRIBUTES id:ID

ELEMENT abandoned (child of segment)
ATTRIBUTES id:ID

ELEMENT self_talk (child of segment)
ATTRIBUTES id:ID

ELEMENT task (child of segment)
ATTRIBUTES id:ID

ELEMENT task_management (child of segment)
ATTRIBUTES id:ID

ELEMENT communication_management (child of segment)
ATTRIBUTES id:ID

ELEMENT statement (child of segment)
ATTRIBUTES id:ID

ELEMENT action_directive (child of segment)
ATTRIBUTES id:ID

ELEMENT info_request (child of segment)
ATTRIBUTES id:ID

ELEMENT offer___commit___open_option (child of segment)
ATTRIBUTES id:ID

ELEMENT opening (child of segment)
ATTRIBUTES id:ID

ELEMENT closing (child of segment)
ATTRIBUTES id:ID

ELEMENT other_forward_function (child of segment)
ATTRIBUTES id:ID

ELEMENT accept (child of segment)
ATTRIBUTES id:ID

ELEMENT maybe___accept_part (child of segment)
ATTRIBUTES id:ID

ELEMENT reject_part (child of segment)
ATTRIBUTES id:ID

ELEMENT reject (child of segment)
ATTRIBUTES id:ID

ELEMENT hold (child of segment)
ATTRIBUTES id:ID

ELEMENT signal_non_understanding (child of segment)
ATTRIBUTES id:ID

ELEMENT acknowledge (child of segment)
ATTRIBUTES id:ID

ELEMENT repeat_rephrase (child of segment)
ATTRIBUTES id:ID

ELEMENT completion (child of segment)
ATTRIBUTES id:ID

ELEMENT correct_misspeaking (child of segment)
ATTRIBUTES id:ID

ELEMENT answer (child of segment)
ATTRIBUTES id:ID

7. **Example:**

Internal structure (DAMSL annotation):

```
<segment id="x4" href="word.xml#id(word_013)">
    <information_level id="il_communication_management__x5">
      <communication_management id="x5"/>
    </information_level>
    <backward_looking_function id="blf_acknowledge__x6">
      <understanding id="u_acknowledge__x6">
        <signal_understanding id="su_acknowledge__x6">
          <acknowledge id="x6"/>
        </signal_understanding>
      </understanding>
    </backward_looking_function>
  </segment>
```

Surface structure (variant of SWBD-DAMSL annotation):

```
  <segment id="x4" href="word.xml#id(word_013)">
```

```
    <communication_management id="x5"/>
    <acknowledge id="x6"/>
  </segment>
```

Here assuming that the word level is in file word.xml and contains the lines:

```
<word id="word_013" who="s">Alright</word>
```

8. **Coding procedure:**

   This procedure describes how to train naive human annotators on the surface scheme from start to ensure relatively homogenous annotations. The description is given on a very general level just to give an impression how annotation and scheme enhancement work.

   1. Introduction of scheme based on the coding book provided.
   2. Training on test corpus.
   3. Evaluation of test corpus annotation.
   4. Revision of scheme.
   5. Repeat 2. to 4. until evaluation shows good results.
   6. Normal annotation with random checks.

9. **Creation notes:**

   Author: Marion Klein.
   Version: 1 (1999/04/21)
   Comment: Serves as an example for "best practice" in the MATE project.
   Literature: [J. Allen & M. Core 97], [D. Jurafsky et al. 97].

1. **Module name:** Map Task dialogue acts [MT-DA].
2. **Module purpose:** Dialogue act coding for HCRC Map Task Dialogues.
3. **Coding level:** Dialogue acts.
4. **Data source:**HCRC Map Task Corpus and DCIEM Map Task Corpus. However, this coding module can be used on other versions of the Map Task, sometimes with small modifications to fit behaviours in different languages.
5. **Module references:** Timed-unit level transcriptions (words plus silences and noises).
6. **Coding Structure:**

   A dialogue is segmented into dialogue moves and inter-move-silences. Each dialogue move consists of one or more words, silences or noises, and is labelled with its move type (see below for list of move types) and each inter-move-silence consists of one or

more silences or noises. Separate coding is done for each speaker. A move inherits its start and end times from the words contained in it, so start and end times are not marked explicitly at the move annotation level.

There are thirteen move types:

> instruct, explain, query-yn, query-w, check, align, reply-y, reply-n, reply-w, acknowledge, clarify, unclassifiable.

For full details of the XML markup, see the dtds:

Dialogue Moves dtd
Timed Units dtd
Word Base dtd which the two dtds above refer to.

7. **Example:**

This is part of a move file for the first speaker

```
<move id="q1ec1.g.move.1" who="giver" label="ready"
href="&gfile;#id(q1ec1g.1)"/>

<move id="q1ec1.g.move.2" who="giver" label="instruct"
href="&gfile;#id(q1ec1g.4)..id(q1ec1g.14)"/>

<ims id="q1ec1.g.move.3.5" who="giver"
href="&gfile;#id(q1ec1g.16)..id(q1ec1g.19)"/>
```

It refers to this timed-unit file for the first speaker

```
<tu id="q1ec1g.1" start="0.0000" end="0.3294" utt="1">okay</tu>
<tu id="q1ec1g.4" start="0.3294" end="0.8432" utt="1">starting</tu>
<tu id="q1ec1g.5" start="0.8432" end="1.3702" utt="1">off</tu>
<sil id="q1ec1g.6" start="1.3702" end="1.5777"/>
<tu id="q1ec1g.7" start="1.5777" end="1.8413" utt="1">we</tu>
<tu id="q1ec1g.8" start="1.8414" end="2.2201" utt="1">are</tu>
<sil id="q1ec1g.9" start="2.2201" end="2.3518"/>
<tu id="q1ec1g.10" start="2.3518" end="2.8722" utt="1">above</tu>
<sil id="q1ec1g.11" start="2.8722" end="2.9644"/>
<tu id="q1ec1g.12" start="2.9644" end="3.0369" utt="1">a</tu>
<tu id="q1ec1g.13" start="3.0369" end="3.5244" utt="1">caravan</tu>
<tu id="q1ec1g.14" start="3.5244" end="3.9394" utt="1">park</tu>
<noi id="q1ec1g.16" start="3.9394" end="4.2885" type="nonvocal"/>
<sil id="q1ec1g.17" start="4.2885" end="4.5784"/>
<noi id="q1ec1g.18" start="4.5784" end="4.8617" type="lipsmack"/>
<noi id="q1ec1g.19" start="4.8617" end="5.3492" type="breath"/>
```

8. **Coding procedure:**

Coders have typically been linguistics or cognitive science postgraduate students, but do not necessarily have to be. Training proceeds in two phases: first the coders read the entire coding manual, with the opportunity to ask questions, and annotate a few dialogues, concentrating on the decision tree page of the manual. Then they discuss

their coding with an experienced annotator. Training can be completed in a day. Coders then begin work in earnest. All coders recode some of their earliest dialogues at the end of their work so that coding stability can be tested, and code some dialogues which other coders have already completed, so that reproducibility can be tested. Coders vary considerably in speed but typically can complete four dialogues per working day. Coders should not be employed solely to code but should have this task interspersed with others so that they do not become too bored with their work. Coders may ask questions of an experienced annotator at any time; questions are common for relatively new coders but tail off with time.

9. **Creation notes:**

Authors: Jean Carletta, Amy Isard.
Version: 1 (1999/04/13)
Comment: Created for use by the HCRC Dialogue Group. Serves as an example for the MATE project.
Literature: [Carletta et al. 1997, Carletta et al. 1996].

1. **Module name:** Verbmobil dialogue acts [VM-DA].

2. **Module purpose:** Dialogue act coding for appointment scheduling and hotel reservation dialogues.

3. **Coding level:** Dialogue acts.

4. **Data source:** BAS spoken dialogue corpora for Verbmobil.

5. **Module references:** canonical level/phonetic transcription module.

6. **Markup declaration:**

ELEMENT DAS
ATTRIBUTES id: ID href: HREF(canonical level)

ELEMENT DA (child of DAS)
ATTRIBUTES id:ID da: TEXT (label) direction: TEXT (speaker, listener)

7. **Description:**

da value: "GREET" | "BYE" | "INTRODUCE" | "POLITENESS_FORMULA" | "THANK" | "DELIBERATE" | "BACKCHANNEL" | "INIT" | "DEFER" | "CLOSE" | "REQUEST" | "REQUEST_SUGGEST" | "REQUEST_CLARIFY" | "REQUEST_COMMENT" | "REQUEST_COMMIT" | "SUGGEST" | "INFORM" | "DIGRESS" | "DEVIATE_SCENARIO" | "REFER_TO_SETTING" | "EXCLUDE" | "CLARIFY" | "GIVE_REASON" | "EXPLAINED_REJECT" | "FEEDBACK" |

"FEEDBACK_NEGATIVE" | "REJECT" | "FEEDBACK_POSITIVE" | "ACCEPT" | "CONFIRM" | "COMMIT" | "OFFER" | "NOT_CLASSIFIABLE"

---

direction value: speaker id "," listener id

8. **Example:**

```
<DAS id="e032ach_RGM_DAS_9"
href="e032ach_RGM_KAN.xml#id(e032ach_RGM_TRN5_x0)..id(e032ach_RGM_TRN5_x1)">
  <DA id="e032ach_RGM_DA_9" da="CLARIFY" direction="RGM,DNC"/>
</DAS>
```

Here assuming that the basic transcription is in file e032ach_RGM_KAN.xml and contains the lines:

```
<KAN id="e032ach_RGM_TRN5_x0">Vv</KAN>
<KAN id="e032ach_RGM_TRN5_x1">n@Uvemb=r</KAN>
```

9. **Coding procedure:**

This procedure describes how to train naive human annotators from the start to ensure relatively homogenous annotations. Time estimates are based on student worker contracts with 8 hours per week.

First stage: 1/2 week
Introduction to theoretical concepts (dialogue acts) and motivation (statistical recognition) providing some example annotations.

Second stage: 1/2 week
Annotators observe annotation process of each other together with supervisor. The supervisor explains unintuitive and special cases (e.g. EXPLAINED_REJECT, SUGGEST in negative mode, CLARIFY, GIVE_REASON before argument, etc.), the segmentation process, and introduces specialised terminology, like turn, utterance, and segment.

Third stage: 1 week
- supervised annotation of five selected dialogues
  (supervisor corrects instantly)
- unsupervised annotation with correction of ten dialogues
  (as often as necessary)

Fourth stage:
Normal annotation with random checks. Each annotator annotates different dialogues. Discussion sessions on difficult annotation cases together with supervisor happen twice a month.

10. **Creation notes:**

Authors: Michael Kipp, Marion Klein.
Version: 1 (1999/04/13)

Comment: Created for use in the Verbmobil project. Serves as an example for the MATE project.
Literature: [Alexandersson et al. 1998].

Communication Problems coding Module

Guideline Coding Module

Name: Guidelines.

Coding purpose: Records the different generic and specific guidelines, the violation of which typically leads to communication problems in a dialogue.

Coding level: Communication problems.

Data sources: List of generic and specific guidelines for co-operative dialogue design.

Module references: None.

Markup declaration:

```
ELEMENT aspect
ELEMENT guideline
ATTRIBUTES
aspect: REFERENCE(this, aspect)
gricean: ENUM (yes|no)
subsumed_by: REFERENCE(this, guideline)
abbreviation: TEXT
```

Description: Two elements are used to annotate the guidelines. One is `aspect`. `aspect` is used to indicate a grouping of the guidelines. For example, the 24 guidelines in Figure 1 are divided into seven groups or aspects. The element `aspect` has no explicit attributes. The mandatory attribute `id` which is a unique identifier, is always generated automatically for all elements.

A second element is `guideline` which marks up a particular guideline. `guideline` has four attributes.

`aspect` is mandatory. It is a reference to the aspect to which the guideline belongs. The `aspect` indicated for a specific guideline must always equal the `aspect` indicated for the generic guideline by which it is subsumed.

`gricean` is mandatory for guidelines which are the same as Grice's maxims [Grice 1975]. The `yes` value is used to indicate a maxim. For non-maxims `gricean` is optional. If indicated, the `no` value must be chosen. Using the value `yes` indicates whether a certain guideline is one of Grice's maxims.

`subsumed_by` should always be used for specific guidelines to indicate by which generic guideline it is subsumed. `subsumed_by` cannot be used for generic guidelines.

`abbreviation` is optional but recommended. It provides an abbreviated form of the guideline. It carries the essential meaning and may be easier to remember than the "canonical" expression of the guideline.

Examples:

```
<aspect id="1">Informativeness</aspect>
...
<aspect id="5">Partner asymmetry</aspect>
<guideline id="GG1" aspect="#1" gricean="yes" abbreviation="Say enough">
  Make your contribution as informative as is required (for the current purposes of the exchange).
</guideline>
<guideline id="SG1" aspect="#1" subsumed_by="#GG1" abbreviation="State commitments explicitly">
  Be fully explicit in communicating to users the commitments they have made.
</guideline>
<guideline id="SG2" aspect="#1" subsumed_by="#GG1" abbreviation="Provide immediate feedback">
  Provide feedback on each piece of information provided by the user.
</guideline>
<guideline id="GG2" aspect="#1" gricean="yes" abbreviation="Don't say too much">
  Do not make your contribution more informative then is required.
</guideline>
...
<guideline id="GG10" aspect="#5" abbreviation="Highlight asymmetries">
  Inform the users of important non-normal characteristics which they should take into account
  in order to behave co-operatively in spoken interaction.  Ensure the feasibility of what is
  required of them.
</guideline>
<guideline id="SG4" aspect="#5" subsumed_by="#GG10" abbreviation="State your capabilities">
  Provide clear and comprehensible communication of what the system can and cannot do.
</guideline>
...
```

Coding procedure:

The guidelines for co-operative dialogue design are part of the coding module for communication problems defined below. However, they may also be reused in other coding modules for communication problems. If a user defining a new communication problems module should want to build on a different set of guidelines it may well be that s/he can still reuse the coding module for guidelines defined here. Encoding a set of guidelines using the present coding module is not very complicated and the following procedure is recommended as sufficient:

1. Encode by coder 1.
2. Check by coder 2.

Creation notes:
Authors: Hans Dybkjær and Laila Dybkjær.
Version: 1 (25 November 1998), 2 (19 June 1999).
Comments: None.
Literature: [Bernsen et al. 1998, Dybkjær 1999].

ViolationTypes Coding Module

Name: Violation_types.

Coding purpose: Records the different ways in which generic and specific guidelines are violated in a given corpus, i.e. types of problems found in the corpus. The corpus is implicitly given by a communication problems coding file referring to the problem type coding file as well as to a transcription.

Coding level: Communication problems.

Data sources: List of types of violations of generic and specific guidelines for co-operative dialogue design. The list is generated during analysis of a corpus with respect to communication problems.

Module references: Module Guidelines.

Markup declaration:

```
ELEMENT vtype
ATTRIBUTES
instance_of: REFERENCE(Guidelines, guideline)
alternative_instances: REFERENCE(Guidelines, guideline+)
```

Description: Each description of a violation type is annotated by the element `vtype`. This element has two attributes.

The attribute `instance_of` is mandatory. `instance_of` is a reference to a particular guideline in a file which contains the guidelines for co-operative dialogue.

`alternative_instances` is optional. Guidelines overlap and in some cases the coder may be in doubt whether one or the other guideline was violated. The attribute `alternative_instances` allows the coder to express this doubt by letting him/her indicate one or more (this is what '+' means) other guidelines than the one referred to by `instance_of`.

The body of `vtype` contains the description of the actual type of violation.

Example:

```
<vtype id="SG4-1" instance_of="Guidelines-1999#SG4">
  Too little said on what system can and cannot do: BA often missing;
  time-table enquiries always missing.
</vtype>
```

Coding procedure: Each communication problem is seen as a certain type of violation of a guideline. The violation types are highly task dependent. The file containing these types is built in parallel with the analysis and markup of communication problems. This file is very special in the sense that its contents, i.e. the text, as well as the markup are created at the same time and by the coder. The contents are textual descriptions of the violation types. We recommend to use the same coding procedure for violation types as for markup of communication problems since the two actions are tightly connected. As a minimum the following procedure should be followed:

1. Encode by coders 1 and 2.
2. Check and merge codings (performed by coders 1 and 2 until consensus).

Creation notes:

Authors: Hans Dybkjær and Laila Dybkjær.
Version: 1 (25 November 1998), 2 (19 June 1999).
Comments: None.
Literature: [Bernsen et al. 1998, Dybkjær 1999].

Communication Problems Coding Module

Name: Communication_problems.

Coding purpose: Records the different ways in which generic and specific guidelines are violated in a given corpus. The communication problems coding file refers to a problem type coding file as well as to a transcription.

Coding level: Communication problems.

Data sources: Dialogue corpora.

Module references: Module Basic_orthographic_transcription; Module Violation_types.

Markup declaration:

```
ELEMENT comprob
ATTRIBUTES
  vtype: REFERENCE(Violation_types, vtype)
  wref: REFERENCE(Basic_orthographic_transcription, (w,w)+)
  uref: REFERENCE(Basic_orthographic_transcription, u+)
  caused_by: REFERENCE(this, comprob)
  temp: TEXT
ELEMENT note
ATTRIBUTES
  wref: REFERENCE(Basic_orthographic_transcription, (w,w)+)
  uref: REFERENCE(Basic_orthographic_transcription, u+)
```

Description: In order to annotate communication problems caused by inadequate systems design we use the element `comprob`. It refers to some kind of violation of one of the guidelines listed in Figure 1. The `comprob` element may be used to mark up any part of the dialogue which caused the communication problem. Thus it may be used to annotate one or more words, an entire utterance or even several utterances in which a communication problem was detected. The `comprob` element has five attributes.

The attribute `vtype` is mandatory. `vtype` is a reference to a particular description of a guideline violation in a file which contains the different kinds of violations of the individual guidelines.

Either `wref` or `uref` must be indicated. Both these attributes refer to an orthographic transcription. `wref` delimits the word(s) which caused a communication problem, and `uref` refers to one or more entire utterances which caused a problem.

The attribute `caused_by` is optional. In some cases a communication problem in a dialogue will be caused by a problem which occurred earlier in that dialogue. `caused_by` is used to refer to a communication problem which was found elsewhere in the dialogue and which led to the present communication problem.

`temp` is an optional attribute. It indicates a temporary markup. It usually takes a few dialogues before the coder gets a good grasp of the types of guideline violations which tend to occur in the corpus and what caused them. Often logfile inspection will be needed to make an exact diagnosis. Moreover, some problems become easier to detect when comparing a few dialogues. Thus `temp` is mainly for use during initial markup of a corpus but may also be used later if it is practical to make some temporary notes before making the final diagnosis. The `vtype` attribute overrides whatever communication problems the attribute `temp` indicates

In the beginning of the analysis the `vtype` attribute may be left open and the `temp` attribute filled in to describe the kind of guideline violation identified. Very soon, however, a file containing the violation types should be established and in most cases the `temp` comments can simply be moved to this file and possibly modified to provide a violation type description. Note that due to this and to the coding procedure requiring at least two coders the violation type references in the `vtype` attribute are likely to eventually be re-classified.

The `note` element can be used anywhere in a corpus to comment on whatever the user wants. It refers to one or more words or one or more utterances in the same way as the `comprob` element. The body of the `note` element contains text.

Example:
The following example communication problems markup assumes this snippet of a transcription from the Sundial corpus and refers to the example in the violation types coding module:

```
<u id="S1:7-1-sun" who="S">
  flight information british airways good day can I help you
</u>
<comprob id="3" vtype="Sundial_problems#SG4-1" uref="Sundial#S1:7-1-sun"/>
<note id="2" uref="Sundial#S1:7-1-sun">
  The system provides too little information about its capabilities and limitations.
  It is of course an ideal that little information is necessary.
  However, the risk is that the user will be misled and assume stronger or weaker system
  capabilities than are actually present.  Designers should look out for symptoms to this
  effect.
  The present introduction suggests that users can ask about anything to do with British
  Airways flights.
  No current system is likely to be able to do that.
  Another interpretation of the system's introduction is that it is owned by British Airways
  but can answer any question about flights.  The former interpretation seems the most natural
  one.
  So the system's opening probably should not be deemed ambiguous.
</note>
```

Coding procedure: We recommend to use the same coding procedure for markup of communication problems as for violation types since the two actions are tightly connected. As a minimum the following procedure should be followed:

1. Encode by coders 1 and 2.
2. Check and merge codings (performed by coders 1 and 2 until consensus).

Creation notes:
Authors: Hans Dybkjær and Laila Dybkjær.
Version: 1 (25 November 1998), 2 (19 June 1999).
Comments: For guidance on how to identify communication problems and for a collection of examples the reader is recommended to look at [Dybkjær 1999].
Literature: [Bernsen et al. 1998].

Cross level coding modules

1. **Module name:** Cross-level reference coding module

2. **Module purpose:** Building reference elements to phenomena that are already marked up.

3. **Coding level:** <xr>

4. **Data source:** Existing and marked up levels of description.

5. **Module references:** Existing and marked up levels of description.

6. **Markup declaration:**

Attributes and values of <xlr> elements	
attribute	**values**
id	[ASCII]
href	[ASCII]
who	[ASCII]
cmt	[ASCII]
cert	0 .1 .2 .3 .4 .5 .6 .7 .8 .9 1
stat	draft, reviewed, done
elmname	[ASCII]
restyp	evs, knr
refvar	[ASCII]
props	[ASCII]

1. **Example:**

```
...
...
<xlr
 id="xlr_052"
 href="wordutt.xml#id(w_4413)"
 who="Q4M"
 cmt="first word"
 cert="1"
 stat="done"
 elmname="w"
 restyp="evs"
 refvar="w1"
 props="($d 1^ $w1) and
        ($w1 <>1 $w2) and
        ($w1 # ~ $w2 #) and
        ($w1 # ~ $lw #)"
/>
<xlr
 id="xlr_053"
 href="wordutt.xml#id(w_4414)"
 who="Q4M"
 cmt="second word"
 cert="1"
 stat="done"
 elmname="w"
 restyp="evs"
 refvar="w2"
 props="($d 1^ $w2) and
        ($w1 <>1 $w2) and
        ($w1 # ~ $w2 #)"
/>
<xlr
 id="xlr_054"
 href="lexw.xml#id(lexw_00766)"
 who="Q4M"
 cmt="word type"
```

```
 cert="1"
 stat="done"
 elmname="lexword"
 restyp="knr"
 refvar="lw"
 props="($lw freq >= 500) and
        ($lw 1^ $ll)"
/>
<xlr
 id="xlr_055"
 href="lexf.xml#id(lexf_0487)"
 who="Q4M"
 cmt="word lemma"
 cert="1"
 stat="done"
 elmname="lexlem"
 restyp="knr"
 refvar="ll"
 props="($lw 1^ $ll) and
        ($ll pos ~
         &amp;quot;noun&amp;quot;)"
/>
...
```

2. Coding procedure:

procedural guideline	
o formulate the constraints/the description of the phenomenon in terms of information available. This includes o knowledge on the elements available o knowledge on the attributes and values of the elements available in the data or procedures for the generation of the necessary information on properties o knowledge on the exact constellation of the elements and their properties o the elements involved o their properties and their relations to one another o the combination of individual properties or relations of elements	
Then, there are two options: A standard procedure and an optimized procedure. The optimized procedure aims at fast results.	
standard procedure	**optimized procedure**
o for each element type defined ▪ look for tokens of these elements ▪ check whether they have the properties necessary ▪ check whether they are in the relations to other elements specified ▪ markup those sets of elements that match all the requirements	o for each of the value constraints defined ▪ begin from the type of element which are fewest ▪ for each of the elements found, check the value constraints in an order that starts with those properties which are least likely fulfilled, delete elements from the list whenever they do not match a criterion ▪ continue with the next least frequent type of elements until all value constraints of the elements have been checked o for all of the constraints involving two elements ▪ check the constraints in an order that starts with those which are least likely fulfilled and keep only those elements which match all of the relation criteria o for all of the logical constraints ▪ again check the constraints in an order that starts with those which are least likely fulfilled and keep only those element pairs for which all of the logical constraints are true

	o markup those sets of elements that match the requirements This procedure aims at identifying elements that do not match as early as possible in order to reduce the number of checks.

3. **Creation notes:**

   Author: Andreas Mengel
   Version: 1 (1999/11/25)
   Comment: -
   Literature: -

---

1. **Module name:** Cross-level element coding module

2. **Module purpose:** Encoding of phenomena that can be described be reference to other phenomena that are already marked up.

3. **Coding level:** `<xlnt>`.

4. **Data source:** `<xlr>` elements

5. **Module references:** Cross-level reference coding module.

6. **Markup declaration:**

Attributes and values of `<xlnt>` elements	
attribute	**values**
id	[ASCII]
who	[ASCII]
cmt	[ASCII]
desc	[ASCII]
phename	[ASCII]
cert	0 .1 .2 .3 .4 .5 .6 .7 .8 .9 1
stat	draft, reviewed, done
calc	[ASCII]

1. **Example:**

```
...
<xlnt
 id="xlnt_001"
 href="xlr.xml#id(xlr_001)..id(xlr_004)"
 who="Q4M"
 cmt="why cows?"
 desc="repitions of nouns caused by high token frequency"
 phename="rnhf"
 cert="1"
 stat="done"
 calc="w.freq"
/>
...
<xlnt
 id="xlnt_007"
```

```
 href="xlr.xml#id(xlr_052)..id(xlr_055)"
 who="Q4M"
 cmt="houses"
 desc="repitions of nouns caused by high token frequency"
 phename="rnhf"
 cert="1"
 stat="done"
 calc="w.freq"
/>
...
<xlnt
 id="xlnt_021"
 href="xlr.xml#id(xlr_127)..id(xlr_130)"
 who="Q4M"
 cmt="wall"
 desc="repitions of nouns caused by high token frequency"
 phename="rnhf"
 cert="1"
 stat="done"
 calc="w.freq"
/>
...
```

2. **Coding procedure:**

   Identify elements that satisfy a given description of the phenomenon and mark them up as an <xlnt> element.

3. **Creation notes:**

   Author: Andreas Mengel
   Version: 1 (1999/11/25)
   Comment: -
   Literature: -

---

1. **Module name:** Cross-level document coding module

2. **Module purpose:** Holds together a collection of `<xlnt>` elements.

3. **Coding level:** `<xdct>`.

4. **Data source:** `<xlnt>` elements..

5. **Module references:** Cross-level element coding modue.

6. **Markup declaration:**

Attributes and values of `<xldct>` elements	
attribute	**values**
id	[ASCII]
who	[ASCII]

cmt	[ASCII]
cert	0 .1 .2 .3 .4 .5 .6 .7 .8 .9 1
stat	draft, reviewed, done
srtelm	[ASCII]
srtatt	[ASCII]
srtcrt	str, num
srtord	std, rev
crea	[ASCII]
refnstab	yes, no
reftstab	yes, no

1. **Example:**

```
...
<xldct
 id="xldct_001"
 href="xlnt.xml#id(xlnt_001)..id(xlr_033)"
 who="Q4M"
 cmt="are there semantic issues involved?"
 cert="1"
 stat="done"
 srtelm="lw"
 srtatt="freq"
 srtcrt="num"
 srtord="rev"
 crea="($w1 w)($w2 w)($lw lexw)($ll lexw);
       ($w1 <>1 $w2) and ($w1 # ~ $w2 #) and
       ($w1 # ~ $lw #) and ($lw freq >= 500) and
       ($lw 1^ $ll) and ($ll pos ~ &quot;noun&quot;)"
 refnstab="yes"
 reftstab="yes"
/>
...
```

2. **Coding procedure:**

   Collect valid <xlnt> elements and put them together into one <xldct> element.

3. **Creation notes:**

   Author: Andreas Mengel
   Version: 1 (1999/11/25)
   Comment: -
   Literature: -

Annex 3: DTDs

Prosody

DTD for Layer 1 (Phonetic transcription)

```
<!-- DTD for the MATE project Prosody Level based on D2.1 (june 99)
XML 1.0, XLink 1.0 DTD
18/6/99 -->
<!-- prlayer1 DOCTYPE contains all the element and attribute
     declarations for the layer 1 of the prosody level -->

<!-- Layer 1: phonetic transcription; SAMPA scheme -->
<!-- ABBREVIATIONS -->
<!ENTITY % id
          'id ID #REQUIRED'
>
<!ENTITY % href
          'href      CDATA #IMPLIED
           xml:link  CDATA #FIXED   "simple"
           show      CDATA #FIXED   "embed"
           actuate   CDATA #FIXED   "auto"'
>
<!ENTITY % start
          'start     CDATA #IMPLIED'
>
<!ENTITY % end
          'end       CDATA #IMPLIED'
>

<!-- the values for the stress attribute cannot be written into
     the DTD because the DTD does not allow these characters.
     But the values can be specified in the XML body.
     stress ( " | % )
-->

<!ELEMENT syllable (phone)*>
<!ATTLIST syllable
          %id;
          stress    CDATA  #IMPLIED
          %start;
          %end;>
<!-- the values for the type attribute cannot be written into
     the DTD because the DTD does not allow all of the characters.
     But the values can be specified in the XML body.
     type ( b | c | C | d | D | f | g | G | h | j | k |
            l | K | m | n | J | N | p | r | R | s | S |
            t | T | v | w | x | H | z | Z | ? |
            a | A | { | 6 | Q | O | e | E | @ | 3 | i |
            I | o | 2 | 9 | & | u | U | } | V | y | Y |
            ... | ~ | = | : )
     NB: Combinations of these characters are also possible
         for the representation of diphtongues, long vowels
         or nasalized vowels.
-->
```

```
<!ELEMENT phone    (#PCDATA)>
<!ATTLIST phone
          %id;
          type      CDATA     #IMPLIED
          %start;
          %end;>
```

DTD for Layer 2a (Phonetic representation of iintonation - IPO scheme)

```
<!-- DTD for the MATE project Prosody Level based on D2.1 (june 99)
XML 1.0, XLink 1.0 DTD
18/6/99 -->
<!-- layer2a DOCTYPE contains all the element and attribute declarations
     for the layer 2a of the prosody level -->

<!-- Layer 2: phonetic representation of intonation -->
<!-- Layer 2a: IPO scheme -->
<!-- ABBREVIATIONS -->
<!ENTITY % id
          'id ID #REQUIRED'
>
<!ENTITY % href
          'href      CDATA #IMPLIED
           xml:link  CDATA #FIXED   "simple"
           show      CDATA #FIXED   "embed"
           actuate   CDATA #FIXED   "auto"'
>
<!ENTITY % start
          'start     CDATA #IMPLIED'
>
<!ENTITY % end
          'end       CDATA #IMPLIED'
>

<!ELEMENT f0      (#PCDATA)>
<!ATTLIST f0
          %id;
          value CDATA   #IMPLIED
          %start;
          %end;>

<!ELEMENT closecopy (#PCDATA)>
<!ATTLIST closecopy
          %id;
          value    CDATA   #IMPLIED
          %start;
          %end;
          %href;>
<!-- the values for the type attribute cannot be written into
      the DTD because the DTD does not allow all of the characters.
      But the values can be specified in the XML body.
      type     (0 | Ø | 1 | 2 | 3 | 4 | 5 | A | B |
                C | D | E | &2 | &3 | &4 | &A | &C | &D)
 -->
<!ELEMENT pitmove (#PCDATA)>
<!ATTLIST pitmove
          %id;
          type     CDATA   #IMPLIED
          %start;
          %end;
          %href;>
```

DTD for Layer 2b (Phonetic representation of intonation - INTSINT scheme)

```
<!-- DTD for the MATE project Prosody Level based on D2.1 (june 99)
XML 1.0, XLink 1.0 DTD
18/6/99 -->
<!-- layer2b DOCTYPE contains all the element and attribute declarations
     for the layer 2b of the prosody level -->

<!-- Layer 2: phonetic representation of intonation -->
<!-- Layer 2b: INTSINT scheme -->
<!-- ABBREVIATIONS -->
<!ENTITY % id
          'id ID #REQUIRED'
>
<!ENTITY % href
          'href      CDATA #IMPLIED
           xml:link  CDATA #FIXED   "simple"
           show      CDATA #FIXED   "embed"
           actuate   CDATA #FIXED   "auto"'
>
<!ENTITY % start
          'start     CDATA #IMPLIED'
>
<!ENTITY % end
          'end       CDATA #IMPLIED'
>

<!ELEMENT momel (#PCDATA)>
<!ATTLIST momel
          %id;
          value    CDATA   #IMPLIED
          %start;
          %end;
          %href;>

<!ELEMENT intone (#PCDATA)>
<!ATTLIST intone
          %id;
          type     (T | M | B | H | S | L | U | D) #IMPLIED
          %start;
          %end;
          %href;>
```

DTD for Layer 3 (Phonological representation of intonation - ToBI scheme)

```
<!-- DTD for the MATE project Prosody Level based on D2.1 (june 99)
XML 1.0, XLink 1.0 DTD
18/6/99 -->

<!-- prlayer3 DOCTYPE contains all the element and attribute declarations
     for the layer 3 of the prosody level -->

<!-- Layer 3: phonological representation of intonation; ToBI scheme -->
<!-- ABBREVIATIONS -->
<!ENTITY % id
          'id ID #REQUIRED'
>
<!ENTITY % href
          'href      CDATA #IMPLIED
           xml:link  CDATA #FIXED   "simple"
           show      CDATA #FIXED   "embed"
           actuate   CDATA #FIXED   "auto"'
>
<!ENTITY % start
          'start     CDATA #IMPLIED'
>
<!ENTITY % end
          'end       CDATA #IMPLIED'
>
<!-- the values for the type attribute cannot be written into
       the DTD because the DTD does not allow all of the characters.
       But the values can be specified in the XML body.
       type     (    H*  |   L*  |  L*+H |  L+H* | H+!H* |   L-  |
                     H-  |  !H-  |    L% |    H% |    %H |    *  |
                     *?  |  X*?  |   -   |   -?  |  X-?  |    %  |
                     %?  |  X%?  |  L-L% |  L-H% | H-H%  | H-L%   )
-->
<!ELEMENT tobitone (#PCDATA)>
<!ATTLIST tobitone
```

```
          %id;
          type     CDATA                                #IMPLIED
          class    (pitaccent | phraccent | boundtone) #IMPLIED
          %start;
          %end;
          %href;>

<!ELEMENT target (#PCDATA)>
<!ATTLIST target
          %id;
          type     (EarlyF0 | LateF0) #IMPLIED
          %start;
          %end;
          %href;>

<!ELEMENT f0range (#PCDATA)>
<!ATTLIST f0range
          %id;
          type     CDATA   #FIXED "HiF0"
          %start;
          %end;
          %href;>

<!ELEMENT repair (#PCDATA)>
<!ATTLIST repair
          %id;
          type     CDATA   #FIXED "%r"
          %start;
          %end;
          %href;>
```

DTD for Layer 4 (Prosodic Phrasing - ToBI scheme)

```
<!-- DTD for the MATE project Prosody Level based on D2.1 (june 99)
XML 1.0, XLink 1.0 DTD
18/6/99 -->

<!-- prlayer4 DOCTYPE contains all the element and attribute declarations
     for the layer 4 of the prosody level -->



<!-- Layer 4: prosodic phrasing; ToBI scheme -->
<!-- ABBREVIATIONS -->
<!ENTITY % id
          'id ID #REQUIRED'
>
<!ENTITY % href
          'href      CDATA #IMPLIED
           xml:link  CDATA #FIXED   "simple"
           show      CDATA #FIXED   "embed"
           actuate   CDATA #FIXED   "auto"'
>
<!ENTITY % start
          'start     CDATA #IMPLIED'
>
<!ENTITY % end
          'end       CDATA #IMPLIED'
>

<!-- the values for the type attribute cannot be written into
        the DTD because the DTD does not allow all of the characters.
        But the values can be specified in the XML body.
        type    (  0  |  1- |  1  |  1p | 1p? |  2- |  2  | 2p |
                   2p? |  3- |  3  |  3p | 3p? |  4- |  4  |  X  )
 -->

<!ELEMENT breakindex (#PCDATA)>
<!ATTLIST breakindex
          %id;
          type     CDATA   #IMPLIED
          %start;
          %end;
          %href;>
```

Morphosyntax

DTD for the Edited Transcription Level

```
<!-- edit.dtd -->
<!-- This the DTD for morphosyntactic annotation at the Edited Transcription level -->
<!-- by Claudia Soria -->
<!-- last modification: 02.6.1999 -->
<!-- ABBREVIATIONS -->
<!ENTITY % id.attr
          'id ID #REQUIRED'
>
<!ENTITY % href.attr
          'href      CDATA #IMPLIED
           xml:link  CDATA #FIXED "simple"
           show      CDATA #FIXED "embed"
           actuate   CDATA #FIXED "auto"'
>
<!-- EDITED TRANSCRIPTION FILE LABEL -->
<!ELEMENT edit_file (seg+, dys+)>
<!-- BASIC UNIT LABELS: core scheme -->
<!ELEMENT seg EMPTY>
<!ATTLIST seg
          type     (broken | sic | gap | scomp | ocomp) #REQUIRED
          rep      CDATA                                #REQUIRED
          ins      CDATA                                #REQUIRED
          %id.attr;
          %href.attr;>
<!--Recommended extensions to the core scheme -->
<!ELEMENT dys (repair?, reparandum?, signal?)>
<!ATTLIST dys
          type     CDATA                                #REQUIRED
          %id.attr;
          %href.attr;>
<!-- Optional extensions -->
<!ELEMENT reparandum EMPTY>
<!ATTLIST reparandum
          %id.attr;
          %href.attr;>
<!ELEMENT signal EMPTY>
<!ATTLIST signal
          %id.attr;
          %href.attr;>
<!ELEMENT repair EMPTY>
<!ATTLIST repair
          %id.attr;
          %href.attr;>
```

DTD for the Morphological Word Level

```
<!-- mword.dtd -->
<!-- This is the DTD for morphosyntactic annotation at the morphological word level -->
<!-- by Claudia Soria -->
<!-- last modification: 27.10.1999 -->
<!-- ABBREVIATIONS -->
<!ENTITY % id.att
          'id ID #REQUIRED'
>
<!ENTITY % href.att
          'href     CDATA #IMPLIED
           xml:link CDATA #FIXED "simple"
           show     CDATA #FIXED "embed"
           actuate  CDATA #FIXED "auto"'
>
<!-- MORPHOLOGICAL WORD FILE LABEL -->
```

```
<!ELEMENT mword_file (mw+, cpw*)>
<!-- BASIC UNIT LABEL -->
<!ELEMENT mw (lexit?, stem?, suffix?, prefix?)>
<!ATTLIST mw
          type     (N|V|AJ|PD|AT|AV|AP|C|NU|I|U|R|F|DM|PU) #REQUIRED
          lemma    CDATA                                   #IMPLIED
          subtype  CDATA                                   #IMPLIED
          broken   (Y|N)                                   #IMPLIED
          %id.att;
          %href.att;>
<!ELEMENT cpw (cpw_h?)>
<!ATTLIST cpw
          type     (N|V|AJ|PD|AT|AV|AP|C|NU|I|U|R|F|DM|PU)     #REQUIRED
          subtype  CDATA                                       #IMPLIED
          broken   (Y|N)                                       #IMPLIED
          %id.att;
          %href.att;>
<!ELEMENT cpw_h EMPTY>
<!ATTLIST cpw_h
          %id.att;
          %href.att;>
<!ELEMENT stem (#PCDATA)>
<!ATTLIST stem
          type (N|V) #REQUIRED
          %id.att;
          %href.att;>
<!ELEMENT suffix (#PCDATA)>
<!ATTLIST suffix
          %id.att;
          %href.att;>
<!ELEMENT prefix (#PCDATA)>
<!ATTLIST prefix
          %id.att;
          %href.att;>
<!-- The following element is used in case there is a reference lexicon in xml format -->
<!ELEMENT lexit EMPTY>
<!ATTLIST lexit
          %id.att;
          %href.att;>
```

DTD for the Chunking Level

```
<!-- chunk.dtd -->
<!-- This is the DTD for morphosyntactic annotation at the chunking level -->
<!-- by Claudia Soria -->
<!-- last modification: 27.10.1999 -->
<!-- ABBREVIATIONS -->
<!ENTITY % id.attr
          'id ID #REQUIRED'
>
<!ENTITY % href.attr
          'href      CDATA     #IMPLIED
           xml:link  CDATA     #FIXED "simple"
           show      CDATA     #FIXED "embed"
           actuate   CDATA     #FIXED "auto"'
>
<!-- CHUNK FILE LABEL -->
<!ELEMENT chunk_file (ch+)>
<!-- BASIC UNIT LABELS -->
<!ELEMENT ch (potgov, intro?, aux?, cop?, modal?, caus?)>
<!ATTLIST ch
          type     (ADJ|PA|ADV|SUBORD|N|P|FV|G|I|PART|Di|ADJ_PART|COORD|U) #REQUIRED
          broken   (Y | N)                                                 #IMPLIED
          %id.attr;
          %href.attr;>
<!ELEMENT potgov EMPTY>
<!ATTLIST potgov
          %id.attr;
          %href.attr;>
<!ELEMENT intro EMPTY>
<!ATTLIST intro
          %id.attr;
          %href.attr;>
<!ELEMENT aux EMPTY>
<!ATTLIST aux
```

```
          %id.attr;
          %href.attr;>
<!ELEMENT cop EMPTY>
<!ATTLIST cop
          %id.attr;
          %href.attr;>
<!ELEMENT modal EMPTY>
<!ATTLIST modal
          %id.attr;
          %href.attr;>
<!ELEMENT caus EMPTY>
<!ATTLIST caus
          %id.attr;
          %href.attr;>
```

DTD for the Functional Level

```
<!-- funct.dtd -->
<!-- This is the DTD for morphosyntactic annotation at the functional level -->
<!-- by Claudia Soria -->
<!-- last modification: 27.10.1999 -->
<!-- ABBREVIATIONS -->
<!ENTITY % id.attr
          'id   ID   #REQUIRED'
>
<!ENTITY % href.attr
          'href      CDATA     #IMPLIED
           xml:link  CDATA     #FIXED "simple"
           show      CDATA     #FIXED "embed"
           actuate   CDATA     #FIXED "auto"'
>
<!-- FUNCTIONAL FILE LABEL -->
<!ELEMENT funct_file (funct+)>
<!-- BASIC UNIT LABELS -->
<!ELEMENT funct (head, dep+)>
<!ATTLIST funct
          %id.attr;>
<!ELEMENT head EMPTY>
<!ATTLIST head
          head      CDATA                              #IMPLIED
          diath     (active|passive|middle)            #IMPLIED
          person    (1|2|3)                            #IMPLIED
          number    (sg|pl)                            #IMPLIED
          gender    (m|f|n)                            #IMPLIED
          v_type    (impers)                           "impers"
          %id.attr;
          %href.attr;>
<!ELEMENT dep EMPTY>
<!ATTLIST dep
          type       (subj|dobj|obj2|iobj|mod|comp)    #REQUIRED
          intro       CDATA                            #IMPLIED
          case        CDATA                            #IMPLIED
          synt_real   (n_cl|cl|c|x)                    #IMPLIED
          %id.attr;
          %href.attr;>

<!--ADDITIONAL LABELS-->
<!ELEMENT coord (arg+)>
<!ATTLIST coord
          type         (and|or|comma) #IMPLIED
          %id.attr;
          %href.attr;>
<!ELEMENT arg EMPTY>
<!ATTLIST arg
          %id.attr;
          %href.attr;>
<!ELEMENT bind (arg+)>
<!ATTLIST bind
          %id.attr;
          %href.attr;>
```

Dialogue Acts

Internal Schema (DAMSL)

```
<!-- da_internal.dtd -->
<!-- by Marion Klein -->
<!-- last modification: 18.3.1999 -->


<!-- ABBREVIATIONS -->
<!ENTITY % idAttr    "id         ID      #REQUIRED"
>
<!ENTITY % hRefAttr  'href       CDATA   #IMPLIED
                      xml:link   CDATA   #FIXED     "simple"
                      show       CDATA   #FIXED     "embed"
                      actuate    CDATA   #FIXED     "auto"'
>


<!-- INTERNAL DA FILE LABEL -->
<!ELEMENT da_internal_file (segment+)>

<!-- BASIC UNIT LABEL -->
<!ELEMENT segment (communicative_status?, information_level, forward_looking_function*,
                   backward_looking_function*)>
<!ATTLIST segment %idAttr;
                  %hRefAttr;>

<!-- DIMENSION LABELS -->
<!ELEMENT communicative_status (uninterpretable | abandoned | self_talk)>
<!ATTLIST communicative_status
          %idAttr;>
<!ELEMENT information_level (task? | task_management? |
                             communication_management? | other_level?)>
<!ATTLIST information_level
          %idAttr;>
<!ELEMENT forward_looking_function (statement? | influencing_addressee_future_action? |
                                    info_request? | committing_speaker_future_action? |
                                    conventional? | explicit_performative? |
                                    exclamation? | other_forward_function?)>
<!ATTLIST forward_looking_function
          %idAttr;>
<!ELEMENT backward_looking_function (agreement? | understanding? | answer? )>
<!ATTLIST backward_looking_function
          %idAttr;>

<!-- DIALOGUE ACT LABELS -->
<!ELEMENT uninterpretable EMPTY>
<!ATTLIST uninterpretable
          %idAttr;>
<!ELEMENT abandoned EMPTY>
<!ATTLIST abandoned
          %idAttr;>
<!ELEMENT self_talk EMPTY>
<!ATTLIST self_talk
          %idAttr;>
<!ELEMENT task EMPTY>
<!ATTLIST task
          %idAttr;>
<!ELEMENT task_management EMPTY>
<!ATTLIST task_management
          %idAttr;>
<!ELEMENT communication_management EMPTY>
<!ATTLIST communication_management
          %idAttr;>
<!ELEMENT other_level EMPTY>
<!ATTLIST other_level
          %idAttr;>
<!ELEMENT statement (assert? | reassert? | other_statement?)>
<!ATTLIST statement
          %idAttr;>
<!ELEMENT assert EMPTY>
<!ATTLIST assert
          %idAttr;>
<!ELEMENT reassert EMPTY>
<!ATTLIST reassert
          %idAttr;>
<!ELEMENT other_statement EMPTY>
<!ATTLIST other_statement
```

```
          %idAttr;>
<!ELEMENT influencing_addressee_future_action (open_option? | action_directive?)>
<!ATTLIST influencing_addressee_future_action
          %idAttr;>
<!ELEMENT open_option EMPTY>
<!ATTLIST open_option
          %idAttr;>
<!ELEMENT action_directive EMPTY>
<!ATTLIST action_directive
          %idAttr;>
<!ELEMENT info_request EMPTY>
<!ATTLIST info_request
          %idAttr;>
<!ELEMENT committing_speaker_future_action (offer? | commit?)>
<!ATTLIST commiting_speaker_future_action
          %idAttr;>
<!ELEMENT offer EMPTY>
<!ATTLIST offer
          %idAttr;>
<!ELEMENT commit EMPTY>
<!ATTLIST commit
          %idAttr;>
<!ELEMENT conventional (opening? | closing?)>
<!ATTLIST conventional
          %idAttr;>
<!ELEMENT opening EMPTY>
<!ATTLIST opening
          %idAttr;>
<!ELEMENT closing EMPTY>
<!ATTLIST closing
          %idAttr;>
<!ELEMENT explicit_performative EMPTY>
<!ATTLIST explicit_performative
          %idAttr;>
<!ELEMENT exclamation EMPTY>
<!ATTLIST exclamation
          %idAttr;>
<!ELEMENT other_forward_function EMPTY>
<!ATTLIST other_forward_function
          %idAttr;>
<!ELEMENT agreement (accept? | accept_part? | maybe? | reject_part? | reject? | hold?)>
<!ATTLIST agreement
          %idAttr;>
<!ELEMENT accept EMPTY>
<!ATTLIST accept
          %idAttr;>
<!ELEMENT accept_part EMPTY>
<!ATTLIST accept_part
          %idAttr;>
<!ELEMENT maybe EMPTY>
<!ATTLIST maybe
          %idAttr;>
<!ELEMENT reject_part EMPTY>
<!ATTLIST reject_part
          %idAttr;>
<!ELEMENT reject EMPTY>
<!ATTLIST reject
          %idAttr;>
<!ELEMENT hold EMPTY>
<!ATTLIST hold
          %idAttr;>
<!ELEMENT understanding (signal_non_understanding? | signal_understanding? |
                         correct_misspeaking?)>
<!ATTLIST understanding
          %idAttr;>
<!ELEMENT signal_non_understanding EMPTY>
<!ATTLIST signal_non_understanding
          %idAttr;>
<!ELEMENT signal_understanding (acknowledge? | repeat_rephrase? | completion?)>
<!ATTLIST signal_understanding
          %idAttr;>
<!ELEMENT acknowledge EMPTY>
<!ATTLIST acknowledge
          %idAttr;>
<!ELEMENT repeat_rephrase EMPTY>
<!ATTLIST repeat_rephrase
          %idAttr;>
<!ELEMENT completion EMPTY>
<!ATTLIST completion
          %idAttr;>
<!ELEMENT correct_misspeaking EMPTY>
<!ATTLIST correct_misspeaking
          %idAttr;>
```

```
<!ELEMENT answer EMPTY>
<!ATTLIST answer
          %idAttr;>
```

Surface Schema (SWBD-DAMSL variant)

```
<!-- da-surface.dtd -->
<!-- by Marion Klein -->
<!-- last modification: 19.3.1999 -->

<!-- ABBREVIATIONS -->
<!ENTITY % idAttr    "id          ID      #REQUIRED">
<!ENTITY % hRefAttr  'href        CDATA   #IMPLIED
                      xml:link    CDATA   #FIXED     "simple"
                      show        CDATA   #FIXED     "embed"
                      actuate     CDATA   #FIXED     "auto"'
>


<!-- SURFACE DA FILE LABEL -->
<!ELEMENT da_surface_file (segment+)>


<!-- BASIC UNIT LABEL -->
<!ELEMENT segment (uninterpretable?, abandoned?, self_talk?, task?, task_management?,
                   communication_management?, statement?, action_directive?,
                   info_request?, offer__commit__open_option?, opening?, closing?,
                   other_forward_function?, accept?, maybe__accept_part?, reject_part?,
                   reject?, hold?, signal_non_understanding?, acknowledge?,
                   repeat_rephrase?, completion?, correct_misspeaking?, answer?)>
<!ATTLIST segment
          %idAttr;
          %hRefAttr;>

<!-- DIALOGUE ACT LABELS -->
<!ELEMENT uninterpretable EMPTY>
<!ATTLIST uninterpretable
          %idAttr;>
<!ELEMENT abandoned EMPTY>
<!ATTLIST abandoned
          %idAttr;>
<!ELEMENT self_talk EMPTY>
<!ATTLIST self_talk
          %idAttr;>
<!ELEMENT task EMPTY>
<!ATTLIST task
          %idAttr;>
<!ELEMENT task_management EMPTY>
<!ATTLIST task_management
          %idAttr;>
<!ELEMENT communication_management EMPTY>
<!ATTLIST communication_management
          %idAttr;>
<!ELEMENT statement EMPTY>
<!ATTLIST statement
          %idAttr;>
<!ELEMENT action_directive EMPTY>
<!ATTLIST action_directive
          %idAttr;>
<!ELEMENT info_request EMPTY>
<!ATTLIST info_request
          %idAttr;>
<!ELEMENT offer__commit__open_option EMPTY>
<!ATTLIST offer__commit__open_option
          %idAttr;>
<!ELEMENT opening EMPTY>
<!ATTLIST opening
          %idAttr;>
<!ELEMENT closing EMPTY>
<!ATTLIST closing
          %idAttr;>
<!ELEMENT other_forward_function EMPTY>
<!ATTLIST other_forward_function
          %idAttr;>
<!ELEMENT accept EMPTY>
<!ATTLIST accept
          %idAttr;>
```

```
<!ELEMENT maybe__accept_part EMPTY>
<!ATTLIST maybe__accept_part
          %idAttr;>
<!ELEMENT reject_part EMPTY>
<!ATTLIST reject_part
          %idAttr;>
<!ELEMENT reject EMPTY>
<!ATTLIST reject
          %idAttr;>
<!ELEMENT hold EMPTY>
<!ATTLIST hold
          %idAttr;>
<!ELEMENT signal_non_understanding EMPTY>
<!ATTLIST signal_non_understanding
          %idAttr;>
<!ELEMENT acknowledge EMPTY>
<!ATTLIST acknowledge
          %idAttr;>
<!ELEMENT repeat_rephrase EMPTY>
<!ATTLIST repeat_rephrase
          %idAttr;>
<!ELEMENT completion EMPTY>
<!ATTLIST completion
          %idAttr;>
<!ELEMENT correct_misspeaking EMPTY>
<!ATTLIST correct_misspeaking
          %idAttr;>
<!ELEMENT answer EMPTY>
<!ATTLIST answer
          %idAttr;>
```

Coreference

```
<!-- DTD for MATE referring expressions                          -->
<!-- Author: Massimo Poesio                                      -->
<!-- Created:  11.06.98, MP                                     -->
<!--           23.08.99, Modified for XML by David McKelvie     -->
<!--           23.08.98, Modified to agree with latest spec of  -->
<!--                     MATE scheme by MP                      -->
<!--           30.11.99, by Amy Isard and MP                    -->

<!ENTITY % hrefAttr 'href       CDATA       #REQUIRED
                     xml:link   CDATA       #FIXED "simple"
                     show       CDATA       #FIXED "embed"
                     actuate    CDATA       #FIXED "auto"'>

<!-- General structure of coref documents -->
<!ELEMENT doc     (head?,coref:universe*,body)>
<!ATTLIST doc
          id      ID      #IMPLIED>
<!ELEMENT head    (title?,author*,note*,date?)>
<!ELEMENT title   (#PCDATA)>
<!ELEMENT author  (#PCDATA)>
<!ELEMENT date    (#PCDATA)>
<!ELEMENT body  (coref:de | coref:link | coref:seg |turn)+>
<!-- Elements having to do with coreference annotation -->
<!-- CORE SCHEME  -->
<!ELEMENT coref:de   ANY>
<!ATTLIST coref:de
                id      ID      #REQUIRED
                type    CDATA   #IMPLIED>

<!ELEMENT coref:link    (coref:anchor)+>
<!ATTLIST coref:link
                %hrefAttr;
                type    (ident | member | subset | poss |
                         e-rel | argptv | prop | bound | f-v | inst |
                         genrel ) #REQUIRED
                subtype (attr | part | sposs | cause ) #IMPLIED
                who-believes CDATA #IMPLIED>

<!ELEMENT coref:anchor  EMPTY>
<!ATTLIST coref:anchor
                %hrefAttr;>
```

```
<!-- UNIVERSES -->
<!ELEMENT coref:universe  (coref:ue)*>
<!ATTLIST coref:universe
               id      ID      #IMPLIED
               modifies (common)       "common">
<!ELEMENT coref:ue  (#PCDATA)>
<!ATTLIST coref:ue
               id  ID #REQUIRED>
<!--                                    OTHER MARKABLES -->

<!ELEMENT coref:seg  (#PCDATA | pause | note | coref:de | coref:seg |
                      turn | u)*>
<!ATTLIST coref:seg
               id      ID      #IMPLIED
               type    CDATA   #IMPLIED>
<!-- Structure of dialogue -->

<!ELEMENT turn  (u|pause|coref:link)*>
<!ATTLIST turn  id ID #REQUIRED
               who CDATA #REQUIRED>
<!ELEMENT u  (#PCDATA | coref:seg | note | coref:de | coref:link)*>
<!ATTLIST u id ID #REQUIRED
           who CDATA #IMPLIED>

<!ELEMENT pause  (#PCDATA)>
<!ATTLIST pause dur CDATA #IMPLIED>

<!ELEMENT note  (#PCDATA)>
<!ATTLIST note
          place CDATA #IMPLIED
          type  CDATA #IMPLIED>
```

Communication Problems

DTD for Guidelines

guidelines.dtd

```
<!-- DTD for Aspects and Guidelines -->
<!-- "guidelines.dtd" -->
<!-- by Amanda Schiffrin, NIS Laboratory, Odense University -->
<!-- last modified 18/6/1999 -->

<!-- ABBREVIATIONS -->
<!ENTITY % idAttr
          'id         ID          #REQUIRED'
>

<!-- INTERNAL GUIDELINES CODING MODULE FILE LABEL - ROOT ELEMENT -->
<!ELEMENT guidelines (aspect*,guideline*)>

<!-- ASPECT DEFINITIONS -->
<!ELEMENT aspect EMPTY>
<!ATTLIST aspect
          %idAttr;
          name             CDATA       #REQUIRED>

<!-- GUIDELINE DEFINITIONS -->
<!ELEMENT guideline (#PCDATA)>
<!ATTLIST guideline
          %idAttr;
          type             (G|S)       #REQUIRED
          aspect           IDREF       #REQUIRED
          subsumed_by      IDREF       #IMPLIED
          gricean          (yes|no)    "no"
          abbr             CDATA       #IMPLIED>
```

DTD for Communication Problems

comprobs.dtd

```
<!-- DTD for comprobs -->
<!-- "comprobs.dtd" -->
<!-- by Amanda Schiffrin, NIS Laboratory, Odense University -->
<!-- last modified 21/10/1999 -->

<!-- ABBREVIATIONS -->
<!ENTITY % idAttr
          'id         ID          #REQUIRED'
>
<!ENTITY % hrefAttr
          'href       CDATA       #REQUIRED
           xml:link   CDATA       #FIXED "simple"
           show       CDATA       #FIXED "embed"
           actuate    CDATA       #FIXED "auto"'
>

<!-- INTERNAL COMMUNICATION PROBLEMS CODING MODULE FILE LABEL -->
<!ELEMENT comprobs (comprob|note)*>
<!ATTLIST comprobs
          %idAttr;
>
<!ELEMENT comprob (vtref)>
<!ATTLIST comprob
          %idAttr;
          %hrefAttr;
          caused_by      IDREF      #IMPLIED
          temp           CDATA      #IMPLIED
>
<!ELEMENT vtref EMPTY>
<!ATTLIST vtref
          %idAttr;
          %hrefAttr;>
<!ELEMENT note (#PCDATA)>
<!ATTLIST note
          %idAttr;
          %hrefAttr;>
```

DTD for Violation Types

vtypes.dtd

```
<!-- DTD for Guideline Violation Types -->
<!-- "vtypes.dtd" -->
<!-- by Amanda Schiffrin, NIS Laboratory, Odense University -->
<!-- last modified 18/6/1999 -->

<!-- ABBREVIATIONS -->
<!ENTITY % idAttr
          'id         ID          #REQUIRED'
>
<!ENTITY % hrefAttr
          'href       CDATA       #REQUIRED
           xml:link   CDATA       #FIXED "simple"
           show       CDATA       #FIXED "embed"
           actuate    CDATA       #FIXED "auto"'
>

<!-- INTERNAL TYPES CODING MODULE FILE LABEL -->
<!ELEMENT vtypes (vtype*)>
<!ATTLIST vtypes
          %idAttr;>

<!-- TYPE DEFINITIONS -->
<!ELEMENT vtype (#PCDATA)>
<!ATTLIST vtype
          %idAttr;
          %hrefAttr;>
```

Cross Level Markup[1]

DTD for Alternatives

altern.dtd

```
<!ELEMENT alt (orig|#PCDATA?)>
<!ATTLIST alt
          id        ID                  #REQUIRED
          lack      (info|theory|agree) #IMPLIED
          who       CDATA               #IMPLIED>
<!ELEMENT orig EMPTY>
<!ATTLIST orig
          id        ID                  #REQUIRED
          href      CDATA               #IMPLIED
          xml:link  CDATA               #FIXED    "simple"
          show      CDATA               #FIXED    "embed"
          actuate   CDATA               #FIXED    "auto">
```

(corresponding example: altern.xml)

DTD for Comments

comment.dtd

```
<!ELEMENT cmnt (#PCDATA)>
<!ATTLIST cmnt
          id        ID     #REQUIRED
          href      CDATA  #IMPLIED
          xml:link  CDATA  #FIXED    "simple"
          show      CDATA  #FIXED    "embed"
          actuate   CDATA  #FIXED    "auto">
```

(corresponding example: comment.xml)

DTD for Definitions

def.dtd

```
<!ELEMENT defs (element)*>
<!ELEMENT element (attribute|#PCDATA)*>
<!ATTLIST element
          id        ID     #REQUIRED
          name      CDATA  #IMPLIED>
<!ELEMENT attribute (value|#PCDATA)*>
<!ATTLIST attribute
          id        ID     #REQUIRED
          name      CDATA  #IMPLIED>
<!ELEMENT value (#PCDATA)>
<!ATTLIST value
          id        ID     #REQUIRED
          name      CDATA  #IMPLIED>
```

(corresponding example: def.xml)

[1] The DTDs are listed in alphabetical order. Note that in the DTDs only those values of attributes are defined that are actually used by the examples they are provided for.

DTD for Definition Links

deflink.dtd

```
<!ELEMENT deflink (used)*>
<!ATTLIST deflink
          id        ID     #REQUIRED
          href      CDATA  #IMPLIED
          xml:link  CDATA  #FIXED    "simple"
          show      CDATA  #FIXED    "embed"
          actuate   CDATA  #FIXED    "auto">
<!ELEMENT used      EMPTY>
<!ATTLIST used
          id        ID     #REQUIRED
          href      CDATA  #IMPLIED
          xml:link  CDATA  #FIXED    "simple"
          show      CDATA  #FIXED    "embed"
          actuate   CDATA  #FIXED    "auto">
          name      CDATA  #IMPLIED>
```

(corresponding example: deflink.xml)

DTD for Deletions

delete.dtd

```
<!ELEMENT pho EMPTY>
<!ATTLIST pho
          id        ID     #REQUIRED
          href      CDATA  #IMPLIED
          xml:link  CDATA  #FIXED    "simple"
          show      CDATA  #FIXED    "embed"
          actuate   CDATA  #FIXED    "auto">
```

(corresponding example: delete.xml)

DTD for Discontinous Entities

discont.dtd

```
<!ELEMENT s (wlink)*>
<!ATTLIST s
          id        ID     #REQUIRED>
<!ELEMENT wlink EMPTY>
<!ATTLIST wlink
          id        ID     #REQUIRED
          href      CDATA  #IMPLIED
          xml:link  CDATA  #FIXED    "simple"
          show      CDATA  #FIXED    "replace"
          actuate   CDATA  #FIXED    "auto">
```

(corresponding examples: discont2.xml)

DTDs for Element Examples

e1.dtd

```
<!ELEMENT aelm (celm)>
<!ATTLIST aelm
          id        ID     #REQUIRED>
<!ELEMENT celm EMPTY>
<!ATTLIST celm
          id        ID     #REQUIRED
```

```
          href      CDATA  #IMPLIED
          xml:link  CDATA  #FIXED    "simple"
          show      CDATA  #FIXED    "embed"
          actuate   CDATA  #FIXED    "auto">
```

(corresponding example: E1)

e2.dtd

```
<!ELEMENT celm (delm)+>
<!ATTLIST celm
          id        ID     #REQUIRED>
<!ELEMENT delm EMPTY>
<!ATTLIST delm
          id        ID     #REQUIRED
          href      CDATA  #IMPLIED
          xml:link  CDATA  #FIXED    "simple"
          show      CDATA  #FIXED    "embed"
          actuate   CDATA  #FIXED    "auto">
```

(corresponding example: E2)

e3.dtd

```
<!ELEMENT celm (delm)+>
<!ATTLIST delm
          id        ID     #REQUIRED>
<!ELEMENT delm EMPTY>
<!ATTLIST delm
          id        ID     #REQUIRED
          r         (e|f)  #IMPLIED
          href      CDATA  #IMPLIED
          xml:link  CDATA  #FIXED    "simple"
          show      CDATA  #FIXED    "embed"
          actuate   CDATA  #FIXED    "auto">

                    or

<!ELEMENT celm (eelm|felm)+>
<!ATTLIST celm
          id        ID     #REQUIRED>
<!ELEMENT eelm EMPTY>
<!ATTLIST eelm
          id        ID     #REQUIRED
          href      CDATA  #IMPLIED
          xml:link  CDATA  #FIXED    "simple"
          show      CDATA  #FIXED    "embed"
          actuate   CDATA  #FIXED    "auto">
<!ELEMENT felm EMPTY>
<!ATTLIST felm
          id        ID     #REQUIRED
          href      CDATA  #IMPLIED
          xml:link  CDATA  #FIXED    "simple"
          show      CDATA  #FIXED    "embed"
          actuate   CDATA  #FIXED    "auto">
```

(corresponding example: E3)

DTD for Annotation History

hist.dtd

```
<!ELEMENT pho EMPTY>
<!ATTLIST pho
          id        ID     #REQUIRED
          type      (I)    #IMPLIED
          href      CDATA  #IMPLIED
          xml:link  CDATA  #FIXED    "simple"
          show      CDATA  #FIXED    "embed"
          actuate   CDATA  #FIXED    "auto">
```

(corresponding example: hist.xml)

DTD for Insertions

insert.dtd

```
<!ELEMENT pho EMPTY>
<!ATTLIST pho
          id        ID     #REQUIRED
          type      (I)    #IMPLIED
          href      CDATA  #IMPLIED
          xml:link  CDATA  #FIXED    "simple"
          show      CDATA  #FIXED    "embed"
          actuate   CDATA  #FIXED    "auto">
```

(corresponding example: insert.xml)

DTDs for Lexical Resources

lex.dtd

```
<!ELEMENT lex EMPTY>
<!ATTLIST lex
          id        ID     #REQUIRED
          stem      CDATA  #IMPLIED
          href      CDATA  #IMPLIED
          xml:link  CDATA  #FIXED    "simple"
          show      CDATA  #FIXED    "embed"
          actuate   CDATA  #FIXED    "auto">
```

(corresponding example: lex.xml)

lex2.dtd

```
<!ELEMENT lem (#PCDATA)>
<!ATTLIST lem
          id        ID         #REQUIRED
          pos (noun|det|verb)  #IMPLIED
          num (sg|pl)          #IMPLIED
          pron      CDATA      #IMPLIED
          href      CDATA      #IMPLIED
          xml:link  CDATA      #FIXED    "simple"
          show      CDATA      #FIXED    "embed"
          actuate   CDATA      #FIXED    "auto">
```

(corresponding example: lex2.xml)

lexf.dtd

```
<!ELEMENT lexf (#PCDATA)>
<!ATTLIST lexf
          id        ID     #REQUIRED
          pos       (noun) #IMPLIED
          base      CDATA  #IMPLIED
          href      CDATA  #IMPLIED
          xml:link  CDATA  #FIXED    "simple"
          show      CDATA  #FIXED    "embed"
          actuate   CDATA  #FIXED    "auto">
```

(corresponding example: lexf.xml)

lexw.dtd

```
<!ELEMENT lexw (#PCDATA)>
<!ATTLIST lexw
          id        ID     #REQUIRED
          freq      CDATA  #IMPLIED
```

```
          href      CDATA  #IMPLIED
          xml:link  CDATA  #FIXED    "simple"
          show      CDATA  #FIXED    "embed"
          actuate   CDATA  #FIXED    "auto">
```

(corresponding example: lexw.xml)

DTD for Alternative Sequence of Entities

neworder.dtd

```
<!ELEMENT pho EMPTY>
<!ATTLIST pho
          id        ID     #REQUIRED
          href      CDATA  #IMPLIED
          xml:link  CDATA  #FIXED    "simple"
          show      CDATA  #FIXED    "embed"
          actuate   CDATA  #FIXED    "auto">
```

(corresponding example: neworder.xml)

DTDs for Reference Examples

normpho.dtd

```
<!ELEMENT pho EMPTY>
<!ATTLIST pho
          id        ID        #REQUIRED
          type      (m|I|s|t) #IMPLIED>
```

(corresponding example: normpho.xml)

normw.dtd

```
<!ELEMENT w (#PCDATA)>
<!ATTLIST w
          id        ID     #REQUIRED
          who       (pt)   #IMPLIED
          href      CDATA  #IMPLIED
          xml:link  CDATA  #FIXED    "simple"
          show      CDATA  #FIXED    "embed"
```

(corresponding example: normw.xml)

normword.dtd

```
<!ELEMENT w (#PCDATA)>
<!ATTLIST w
          id        ID     #REQUIRED
```

(corresponding example: normword.xml)

normword2.dtd

```
<!ELEMENT w (#PCDATA)>
<!ATTLIST w
          id        ID     #REQUIRED
```

(corresponding example: normword2.xml)

normsent.dtd

```
<!ELEMENT s EMPTY>
<!ATTLIST s
          id        ID     #REQUIRED
          href      CDATA  #IMPLIED
          xml:link  CDATA  #FIXED    "simple"
          show      CDATA  #FIXED    "embed"
          actuate   CDATA  #FIXED    "auto">
```

(corresponding example: normsent.xml)

pho.dtd

```
<!ELEMENT              pho              EMPTY>
<!ATTLIST                                  pho
          id        ID                 #REQUIRED
          type      (t|r|I|k|w|O:|z) #IMPLIED>
```

(corresponding example: pho.xml)

DTD for Replacements

replace.dtd

```
<!ELEMENT pho EMPTY>
<!ATTLIST pho
          id        ID     #REQUIRED
          type      (U)    #IMPLIED
          href      CDATA  #IMPLIED
          xml:link  CDATA  #FIXED    "simple"
          show      CDATA  #FIXED    "embed"
          actuate   CDATA  #FIXED    "auto">
```

(corresponding example: replace.xml)

DTD for Source Documents

reso.dtd

```
<!ELEMENT project (file)*>
<!ATTLIST project
          id        ID             #REQUIRED
          prjname   CDATA          #IMPLIED>
<!ELEMENT file      EMPTY>
<!ATTLIST file
          id        ID             #REQUIRED
          fenc      (bin|xml|asc)  #IMPLIED>
```

(corresponding example: reso.xml)

DTD for Sentence and Word Example

sentword.dtd

```
<!ELEMENT sent (word)+>
<!ATTLIST sent
          id        ID           #REQUIRED
          type      (ass)        #IMPLIED
```

```
          who       (mary)       #IMPLIED>
<!ELEMENT word (#PCDATA)>
<!ATTLIST w
          id        ID           #REQUIRED
          pos       (ART|NN|V))  #IMPLIED
          num       (sg|pl)      #IMPLIED
          href      CDATA        #IMPLIED
          xml:link  CDATA        #FIXED    "simple"
          show      CDATA        #FIXED    "embed"
          actuate   CDATA        #FIXED    "auto">
```

(corresponding example: sentword.xml)

DTD for Thesauri

thes.dtd

```
<!ELEMENT thesrel (theselm)+>
<!ATTLIST thesrel
          id        ID                                      #REQUIRED
          type (syn|ant|isa)                                #IMPLIED>
<!ELEMENT theselm EMPTY>
<!ATTLIST theselm
          id        ID                                      #REQUIRED
          role      (synonym|broadterm|narrowterm|antonym)  #IMPLIED
          href      CDATA                                   #IMPLIED
          xml:link  CDATA                                  #FIXED    "simple"
          show      CDATA                                   #FIXED    "embed"
          actuate   CDATA                                   #FIXED    "auto">
```

(corresponding example: thes.xml)

theswrong.dtd

```
<!ELEMENT          theswrong          EMPTY>
<!ATTLIST                            theswrong
             id           ID         #REQUIRED
             roles         CDATA      #IMPLIED
             href          CDATA      #IMPLIED
          xml:link  CDATA  #FIXED     "simple"
          show       CDATA  #FIXED     "embed"
          actuate   CDATA  #FIXED     "auto">
```

(corresponding example: theswrong.xml)

thesgood.dtd

```
<!ELEMENT        thesgood        (theselm)+>
<!ATTLIST                          thesgood
          id        ID     #REQUIRED>

<!ELEMENT           theselm           EMPTY>
<!ATTLIST                            theselm
             id           ID         #REQUIRED
             role          CDATA      #IMPLIED
             href          CDATA      #IMPLIED
          xml:link  CDATA  #FIXED     "simple"
          show       CDATA  #FIXED     "embed"
          actuate   CDATA  #FIXED     "auto">
```

(corresponding example: thesgood.xml)

DTDs for Examples

utter.dtd

```
<!ELEMENT w EMPTY>
<!ATTLIST w
          id        ID       #REQUIRED
          num       (sg|pl)  #IMPLIED
          case      (nom)    #IMPLIED
          href      CDATA    #IMPLIED
          xml:link  CDATA    #FIXED    "simple"
          show      CDATA    #FIXED    "embed"
          actuate   CDATA    #FIXED    "auto">
```

(corresponding example: utter.xml)

word.dtd

```
<!ELEMENT w (pho)+>
<!ATTLIST w
          id        ID     #REQUIRED
          href      CDATA  #IMPLIED
          xml:link  CDATA  #FIXED    "simple"
          show      CDATA  #FIXED    "embed"
          actuate   CDATA  #FIXED    "auto">
```

(corresponding example: word.xml)

word2.dtd

```
<!ELEMENT w (lexspec)>
<!ATTLIST w
          id        ID     #REQUIRED
          href      CDATA  #IMPLIED
          xml:link  CDATA  #FIXED    "simple"
          show      CDATA  #FIXED    "embed"
          actuate   CDATA  #FIXED    "auto">
<!ELEMENT lexspec EMPTY>
<!ATTLIST lexspec
          id        ID     #REQUIRED
          href      CDATA  #IMPLIED
          xml:link  CDATA  #FIXED    "simple"
          show      CDATA  #FIXED    "embed"
          actuate   CDATA  #FIXED    "auto">
```

(corresponding example: word2.xml)

wordlex.dtd

```
<!ELEMENT lemma EMPTY>
<!ATTLIST lemma
          id        ID     REQUIRED
          pos       CDATA  #IMPLIED
          pron      CDATA  #IMPLIED
          orth      CDATA  #IMPLIED>
```

(corresponding example: wordlex.xml)

words.dtd

```
<!ELEMENT word (#PCDATA)>
<!ATTLIST word
          id        ID                  #REQUIRED
          pos       (art|nn|v|adj|adv)  #IMPLIED
          href      CDATA               #IMPLIED
          xml:link  CDATA               #FIXED    "simple"
          show      CDATA               #FIXED    "embed"
          actuate   CDATA               #FIXED    "auto">
```

(corresponding example: word001.xml)

wordutt.dtd

```
<!ELEMENT w (#PCDATA)>
<!ATTLIST w
          id        ID     #REQUIRED
          href      CDATA  #IMPLIED
          xml:link  CDATA  #FIXED    "simple"
          show      CDATA  #FIXED    "embed"
          actuate   CDATA  #FIXED    "auto">
```

(corresponding example: wordutt.xml)

DTD for Cross Level Elements

xl.dtd

```
<!ELEMENT xldct (xlnt)*>
<!ATTLIST xldct
          id        ID                                #REQUIRED
          who       CDATA                             #IMPLIED
          cmt       CDATA                             #IMPLIED
          cert      (0|.1|.2|.3|.4|.5|.6|.7|.8|.9|1)  #IMPLIED
          stat      (draft|reviewed|done)             #IMPLIED
          srtelm    CDATA                             #IMPLIED
          srtatt    CDATA                             #IMPLIED
          srtcrt    [str|num]                         #IMPLIED
          srtord    [std|rev]                         #IMPLIED
          crea      CDATA                             #IMPLIED
          refnstab  (yes|no)                          #IMPLIED
          reftstab  (yes|no)                          #IMPLIED
          href      CDATA                             #IMPLIED
          xml:link  CDATA                            #FIXED    "simple"
          show      CDATA                             #FIXED    "embed"
          actuate   CDATA                             #FIXED    "auto">
<!ELEMENT xlnt (xlr)*>
<!ATTLIST xlnt
          id        ID                                #REQUIRED
          who       CDATA                             #IMPLIED
          cmt       CDATA                             #IMPLIED
          desc      CDATA                             #IMPLIED
          phename   CDATA                             #IMPLIED
          cert      (0|.1|.2|.3|.4|.5|.6|.7|.8|.9|1)  #IMPLIED
          stat      (draft|reviewed|done)             #IMPLIED
          calc      CDATA                             #IMPLIED
          href      CDATA                             #IMPLIED
          xml:link  CDATA                            #FIXED    "simple"
          show      CDATA                             #FIXED    "embed"
          actuate   CDATA                             #FIXED    "auto">
<!ELEMENT xlr (EMPTY)>
<!ATTLIST xlr
          id        ID                                #REQUIRED
          who       CDATA                             #IMPLIED
          cmt       CDATA                             #IMPLIED
          cert      (0|.1|.2|.3|.4|.5|.6|.7|.8|.9|1)  #IMPLIED
          stat      (draft|reviewed|done)             #IMPLIED
          name      CDATA                             #IMPLIED
          restyp    (evs|knr)                         #IMPLIED
          refvar    CDATA                             #IMPLIED
          props     CDATA                             #IMPLIED
          href      CDATA                             #IMPLIED
          xml:link  CDATA                            #FIXED    "simple"
          show      CDATA                             #FIXED    "embed"
          actuate   CDATA                             #FIXED    "auto">
```

(corresponding example: xl.xml)