Towards Corpus Annotation Standards
The MATE Workbench

Laila Dybkjær and Niels Ole Bernsen

Natural Interactive Systems Laboratory
Science Park 10, 5230 Odense M, Denmark
Email: laila@nis.sdu.dk, nob@nis.sdu.dk
Tel: +45 65 50 35 53, +45 65 50 35 44, Fax: +45 63 15 72 24

ABSTRACT

This paper describes the European MATE project and its work towards speech corpus annotation standards. Focus is on the MATE workbench which is thoroughly described and illustrated.

1. INTRODUCTION

The aim of the European Telematics project MATE (Multilevel Annotation Tools Engineering) has been to facilitate the use and reuse of spoken language resources, coding schemes and tools by addressing theoretical issues and implementing practical solutions. MATE was launched in 1998 in response to the increasing need - not least in the area of spoken language dialogue systems research and development - for tools and standardisation efforts in support of efficient markup of spoken dialogue corpora at multiple levels.

The main results of the project are the MATE markup framework which bridges between the theoretical and the practical activities of MATE and is proposed as a standard for the definition and representation of markup for spoken dialogue corpora (Dybkjær and Bernsen 2000b), and the MATE workbench which supports the use of the markup framework. Even if the project has formally come to an end, the project consortium has continued funding for improving the workbench and new versions continue to appear. The newest version of the software can always be downloaded from the MATE web site at http://mate.nis.sdu.dk. Both an executable version and the source code is available, the latter under the GNU open source LGP license.

A discussion forum has been set up at the MATE web site for asking questions and sharing experience on the workbench, and for adding new tools to the MATE workbench to enhance its functionality.

In the following, Section 2 briefly describes the theoretical approach of MATE, Section 3 provides a detailed walkthrough of the MATE workbench, and Section 4 mentions related work.

2. THEORETICAL BACKGROUND

To provide a solid basis for the coding standard to be proposed by MATE, more than sixty existing coding schemes belonging to five different coding levels (i.e. prosody, (morpho-) syntax, co-reference, dialogue acts, and communication problems) and their cross-level interaction were reviewed (Klein et al. 1998). The collected information served as background for establishing the MATE markup framework (Dybkjær et al. 1998, Dybkjær and Bernsen 2000b). In MATE, a coding level is some level of abstraction at which to conceptualise, tag, analyse, and retrieve information inherent in language corpora.

The MATE markup framework is a conceptual model which basically prescribes (i) how files are structured, for instance to enable multi-level annotation, (ii) how tag sets are represented in terms of elements and attributes, and (iii) how to provide essential information on markup, semantics, coding purpose etc.

The central concept of the MATE markup framework is the coding module. A coding module is an extended kind of coding scheme which prescribes what constitutes a coding, including the representation of markup and the relations to other codings. Coding modules incorporate (i), (ii) and (iii) above. Thus, the MATE coding module is a proposal for a standard description of coding schemes.

For each of the above-mentioned five annotation levels and the issues to do with cross-level annotation, one or several of the reviewed coding schemes were adopted as starting-points for the definition, following the MATE markup framework, of best practice coding schemes for implementation in the MATE workbench (Mengel et al. 2000).

3. THE MATE WORKBENCH

The MATE workbench (Isard et al. 1998, Isard et al. 2000, Dybkjær and Bernsen 2000a) is a software tool set which supports the MATE markup framework, incorporates the MATE best practice coding schemes, and enables users to annotate corpora and extract information about annotated corpora via a user-friendly interface. The workbench is in continued development and, although the full functionality and desired user-friendliness has not yet been achieved, there is already considerable interest among colleagues from around the world in using the MATE workbench.

The workbench is implemented in Java. It has been tested on Unix (Solaris) and Windows (NT and 98) but should run on any platform for which Java 1.2 or newer is available. The workbench has a modular architecture which facilitates the addition of new modules and new tool functionality by its users.

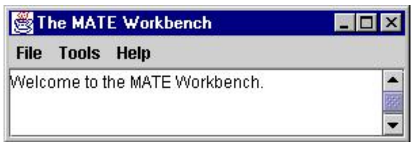
XML is used for internal file representation. Stylesheets are used for specifying the visual presentation of data to users. Stylesheets are written in the MATE Stylesheet Language (MSL). The emerging standard in this area is XSLT. XSLT, however, was not fully defined when the workbench was being designed, and lacked various necessary functionalities at the time. It was therefore decided to implement MSL which uses the MATE query language but is otherwise similar to XSLT.

In the following, we describe and illustrate the MATE workbench functionalities.

3.1 Getting started

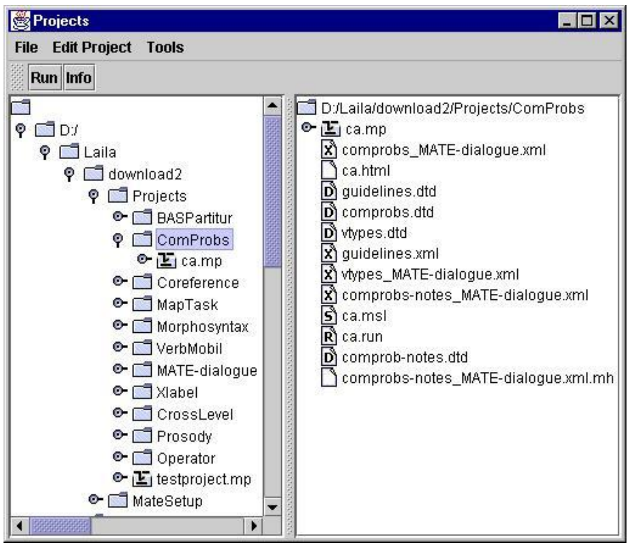
As mentioned, the MATE workbench can be downloaded from the MATE web site at http://mate.nis.sdu.dk. The download page describes how to start the workbench from a command tool. Starting the workbench will result in the following two windows being opened. The first window (Figure 1) allows the user to open new project windows (Figure 2), access available tools, such as the coding module editor (Figure 5), and access the online help function.

The second window (Figure 2) shows example projects which include best practice coding schemes and annotated examples for a number of different annotation levels.



Welcome to the MATE Workbench.

Figure 1. The control window.



```
D:/
  Laila
    download2
      Projects
        BASPartitur
        ComProbs
          ca.mp
        Coreference
        MapTask
        Morphosyntax
        VerbMobil
        MATE-dialogue
        Xlabel
        CrossLevel
        Prosody
        Operator
        testproject.mp
      MateSetup
```
```
D:/Laila/download2/Projects/ComProbs
  ca.mp
  comprobs_MATE-dialogue.xml
  ca.html
  guidelines.dtd
  comprobs.dtd
  vtypes.dtd
  guidelines.xml
  vtypes_MATE-dialogue.xml
  comprobs-notes_MATE-dialogue.xml
  ca.msl
  ca.run
  comprob-notes.dtd
  comprobs-notes_MATE-dialogue.xml.mh
```

Figure 2. The projects window.

3.2 Annotation

The two main functionalities of the MATE workbench are support for annotation of spoken dialogue corpora and linguistic data, and information extraction from annotated corpora. The workbench annotation support is described in the following.

3.2.1 Best practice coding schemes

Ready-to-use best practice coding schemes are included for most of the coding levels addressed by MATE. Prosody is not yet included. Stylesheets allow the user to view and annotate files in an appropriate way using any of the implemented best practice schemes. Example dialogues are provided which illustrate how a tagged dialogue and the accompanying tag set will be shown when a particular best practice scheme is being applied. Internally in the workbench, an annotated dialogue is represented as a set of references to the transcription of the dialogue and possibly to other coding files. By default, the transcription refers to timeline information. The two windows in Figures 3 and 4 show annotation based on the MATE MapTask scheme and the MATE communication problems scheme, respectively. Figures 3 and 4 aptly illustrate the very different display requirements imposed by different coding schemes.

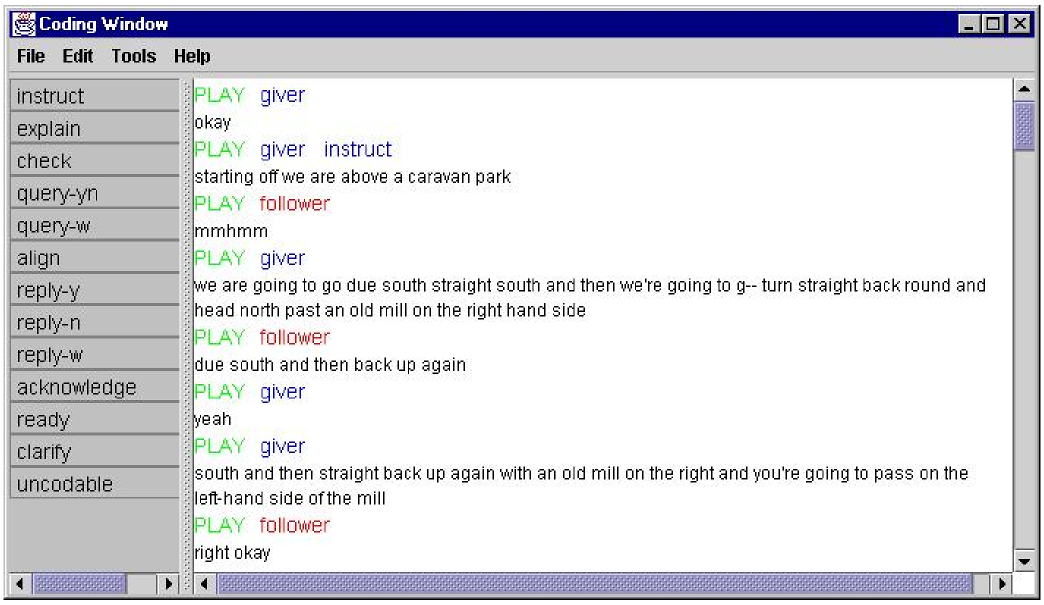
Clicking on the green ‘PLAY’ button in the Map Task window in Figure 3 will result in the audio file corresponding to the transcribed turn being played. The giver and the follower are the two speakers. Each speaker turn can be annotated with a speech act chosen from the list on the left by selecting the speaker and then clicking on the tag to be assigned. For example, the second giver utterance has been annotated with the ‘instruct’ dialogue act.

In the communication problems window in Figure 4 the orthographically transcribed dialogue is shown in the upper left-hand panel. To support the coder, guidelines for cooperative dialogue are shown in abbreviated form in the upper right-hand panel. Types of violation of particular guidelines are incrementally added by the coder in the lower right-hand panel. This panel is empty when annotation starts. The blue markup in the dialogue refers to the types of violation described in the lower right-hand panel and the violations themselves refer to the guidelines. The lower left-hand panel shows annotator's notes. Again, this panel is empty when annotation starts. Notes can be added whenever the annotator needs to add an explanation of, e.g., why something went wrong in the dialogue so as to cause a communication problem.

There is no example coding module for transcription in the MATE workbench. Instead, a converter from Transcriber format (http://www.etca.fr/CTA/gip/Projets/Transcriber/) to MATE format enables transcriptions made using Transcriber to be exported to MATE format and annotated using the MATE workbench.

3.2.2 Adding a new coding module

Users may add new coding modules (coding schemes) themselves for existing or new coding levels via the coding module editor, cf. Figure 5. In order for a coding scheme and the dialogues annotated using it to be usable and understandable by people other than its creator, some key information must be provided. The MATE coding module which is the standard coding scheme description format proposed by MATE, serves to capture this information. A coding module includes the ten items shown in Figure 5. It prescribes what constitutes a coding, including markup representation and relations to other codings (module references).



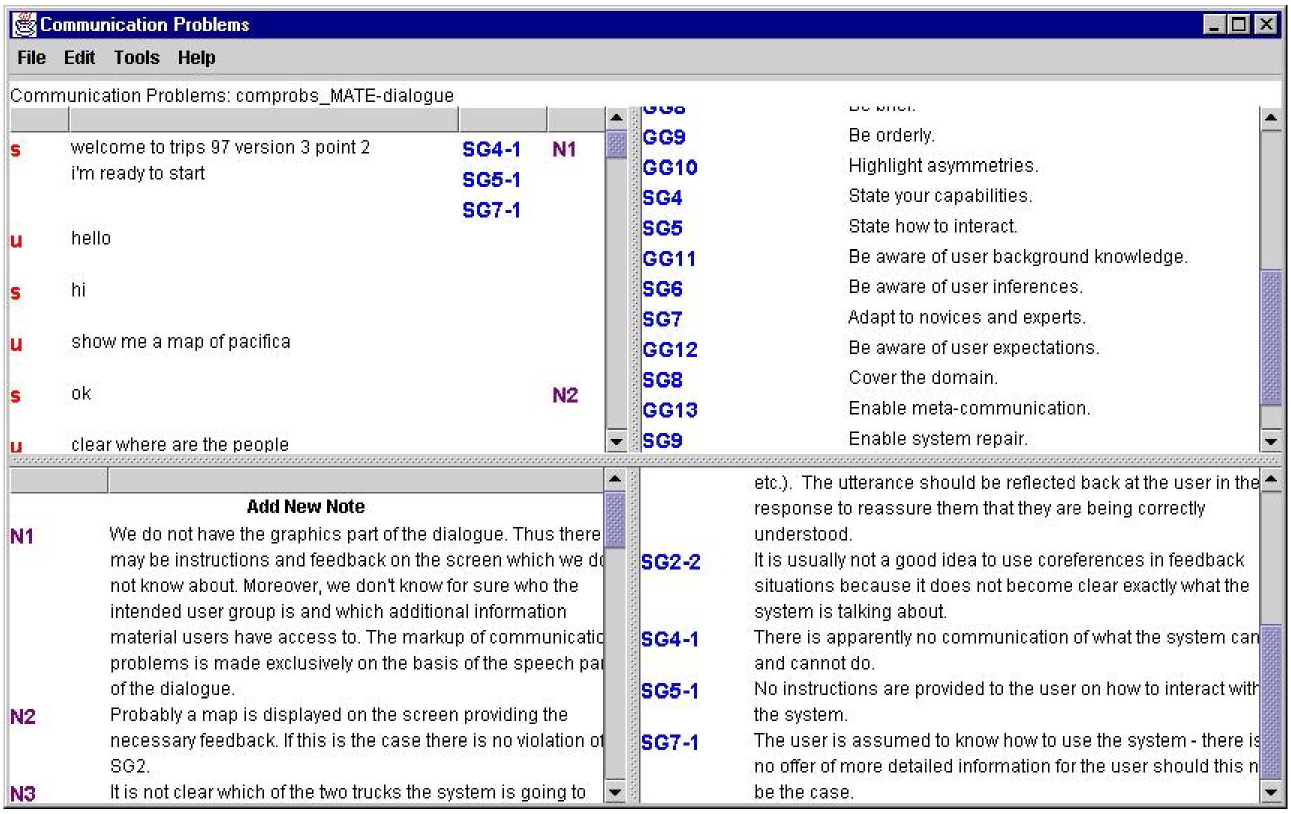
Coding Window

File Edit Tools Help

instruct
explain
check
query-yn
query-w
align
reply-y
reply-n
reply-w
acknowledge
ready
clarify
uncodable

PLAY giver
okay
PLAY giver instruct
starting off we are above a caravan park
PLAY follower
mmhmm
PLAY giver
we are going to go due south straight south and then we're going to g-- turn straight back round and head north past an old mill on the right hand side
PLAY follower
due south and then back up again
PLAY giver
yeah
PLAY giver
south and then straight back up again with an old mill on the right and you're going to pass on the left-hand side of the mill
PLAY follower
right okay

Figure 3. Annotation using the MATE Map Task scheme.



Communication Problems

File Edit Tools Help

Communication Problems: comprobs_MATE-dialogue

s	welcome to trips 97 version 3 point 2 i'm ready to start	SG4-1 SG5-1 SG7-1	N1
u	hello		
s	hi		
u	show me a map of pacifica		
s	ok		N2
u	clear where are the people		

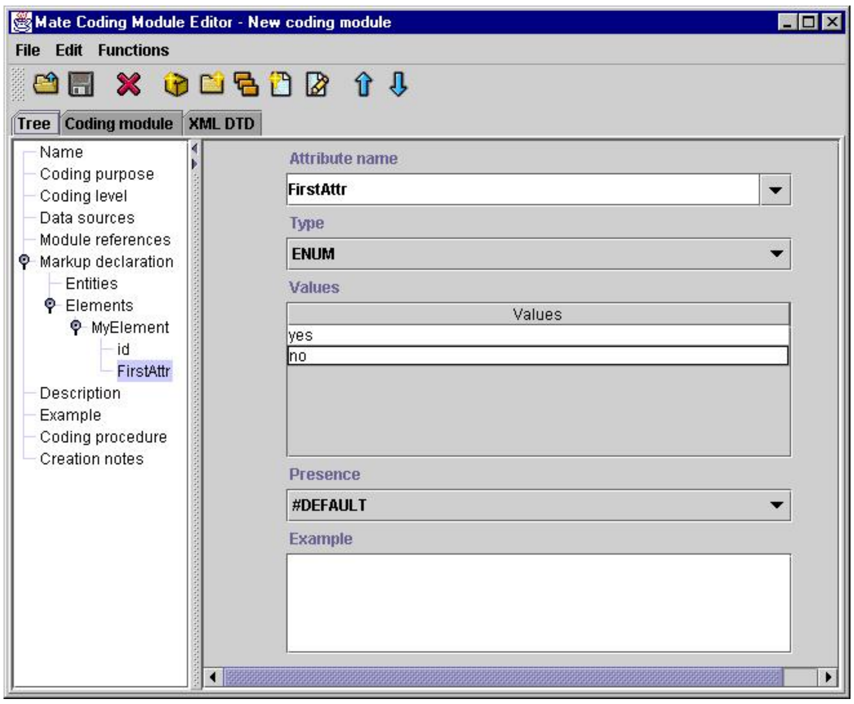
GG9	Be orderly.
GG10	Highlight asymmetries.
SG4	State your capabilities.
SG5	State how to interact.
GG11	Be aware of user background knowledge.
SG6	Be aware of user inferences.
SG7	Adapt to novices and experts.
GG12	Be aware of user expectations.
SG8	Cover the domain.
GG13	Enable meta-communication.
SG9	Enable system repair.

Add New Note

N1	We do not have the graphics part of the dialogue. Thus there may be instructions and feedback on the screen which we d not know about. Moreover, we don't know for sure who the intended user group is and which additional information material users have access to. The markup of communicatio problems is made exclusively on the basis of the speech pa of the dialogue.
N2	Probably a map is displayed on the screen providing the necessary feedback. If this is the case there is no violation o SG2.
N3	It is not clear which of the two trucks the system is going to

	etc.). The utterance should be reflected back at the user in the response to reassure them that they are being correctly understood.
SG2-2	It is usually not a good idea to use coreferences in feedback situations because it does not become clear exactly what the system is talking about.
SG4-1	There is apparently no communication of what the system can and cannot do.
SG5-1	No instructions are provided to the user on how to interact with the system.
SG7-1	The user is assumed to know how to use the system - there is no offer of more detailed information for the user should this n be the case.

Figure 4. Annotation using the MATE communication problems scheme.



Attribute name: FirstAttr

Type: ENUM

Values

Values
yes
no

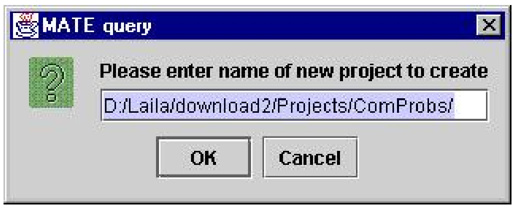
Presence: #DEFAULT

Example:

Figure 5. The coding module editor.

3.2.3 Adding a new project

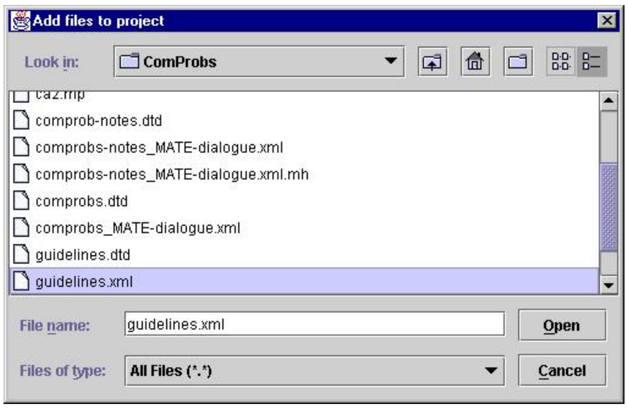
A new MATE project folder can be created (Figure 6) via the 'File' menu in the project window in Figure 2.



Please enter name of new project to create

D:/Laila/download2/Projects/ComProbs/

Figure 6. Creating a new project.



Look in: ComProbs

```
caz.mp
comprob-notes.dtd
comprobs-notes_MATE-dialogue.xml
comprobs-notes_MATE-dialogue.xml.mh
comprobs.dtd
comprobs_MATE-dialogue.xml
guidelines.dtd
guidelines.xml
```

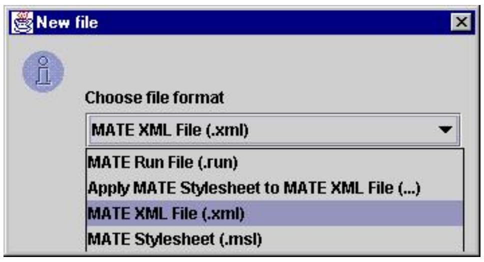
File name: guidelines.xml

Files of type: All Files (*.*)

Figure 7. Browsing to add a file to a project folder.

3.2.4 Adding a new file

Figure 7 shows the browsing for an existing file to be added to a project folder. The window is invoked from the 'Edit Project' menu in the project window in Figure 2. New files in a particular format may be created (Figure 8) via the same menu and files may also be removed via this menu. As a minimum, a new XML file which is being created must be given a name, and the DTD or coding module to be applied must be specified. However, it is recommended that further header information is provided as well, cf. Figure 9.

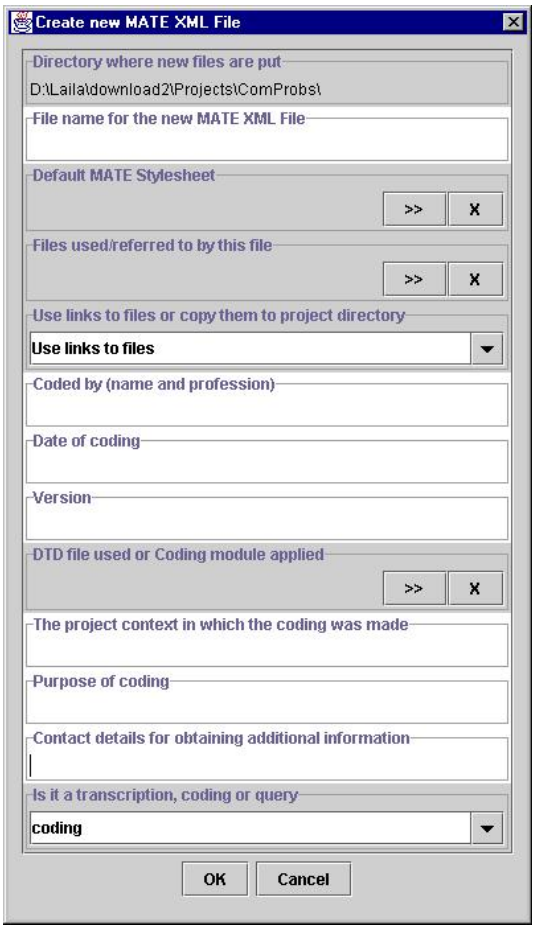


Choose file format: MATE XML File (.xml)

Figure 8. Creating a new file.

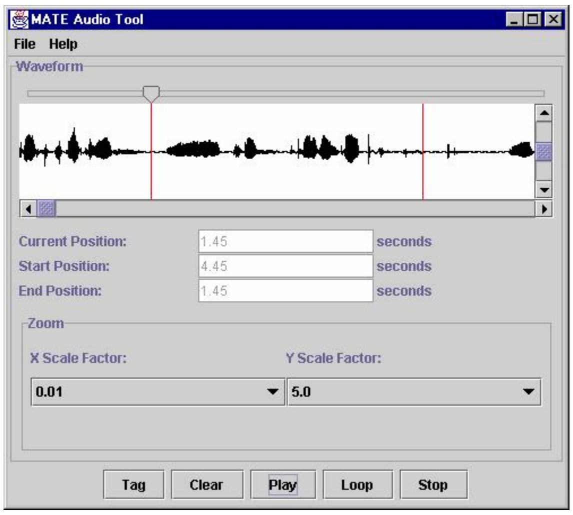
3.2.5 Listening to audio files

Audio files can be selected in the projects window (Figure 2) and played using the MATE audio tool. The window in Figure 10 will then appear displaying the sound file as a waveform. The audio tool might be used during transcription if a user has added an appropriate transcription coding module. As it stands, the audio tool acts as a support tool during annotation and annotation review. For instance, when annotating communication problems there is sometimes a need for listening to the speech file in order to disambiguate an utterance and diagnose what went wrong.



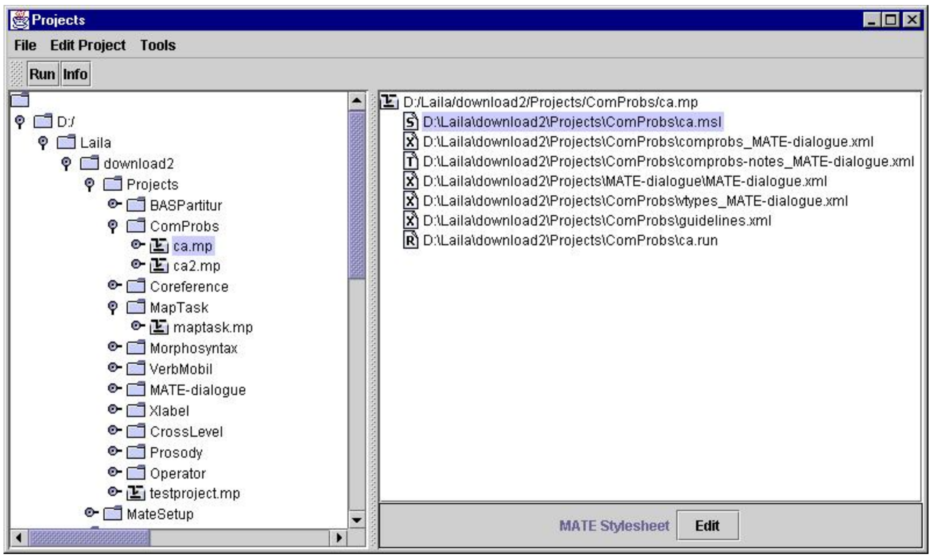
Directory where new files are put: D:\Laila\download2\Projects\ComProbs\
File name for the new MATE XML File:
Default MATE Stylesheet:
Files used/referred to by this file:
Use links to files or copy them to project directory: Use links to files
Coded by (name and profession):
Date of coding:
Version:
DTD file used or Coding module applied:
The project context in which the coding was made:
Purpose of coding:
Contact details for obtaining additional information:
Is it a transcription, coding or query: coding

Figure 9. Creating a new XML file.



Current Position: 1.45 seconds
Start Position: 4.45 seconds
End Position: 1.45 seconds
X Scale Factor: 0.01
Y Scale Factor: 5.0

Figure 10. The MATE audio tool.



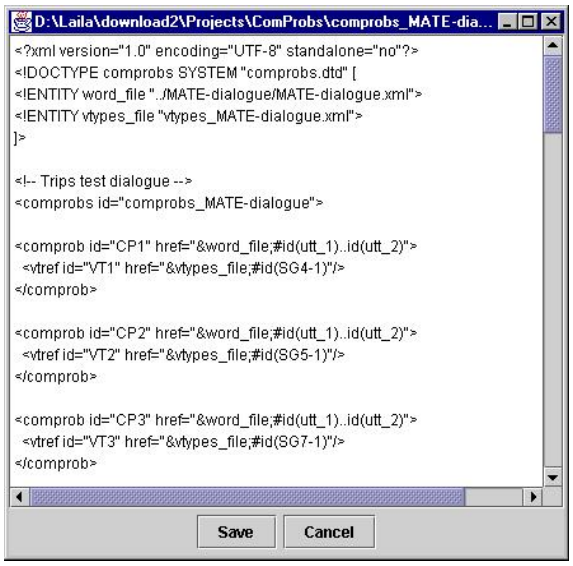
```
D:/
  Laila
    download2
      Projects
        BASPartitur
        ComProbs
          ca.mp
          ca2.mp
        Coreference
        MapTask
          maptask.mp
        Morphosyntax
        VerbMobil
        MATE-dialogue
        Xlabel
        CrossLevel
        Prosody
        Operator
        testproject.mp
      MateSetup
```

```
D:/Laila/download2/Projects/ComProbs/ca.mp
  D:\Laila\download2\Projects\ComProbs\ca.msl
  D:\Laila\download2\Projects\ComProbs\comprobs_MATE-dialogue.xml
  D:\Laila\download2\Projects\ComProbs\comprobs-notes_MATE-dialogue.xml
  D:\Laila\download2\Projects\MATE-dialogue\MATE-dialogue.xml
  D:\Laila\download2\Projects\ComProbs\wtypes_MATE-dialogue.xml
  D:\Laila\download2\Projects\ComProbs\guidelines.xml
  D:\Laila\download2\Projects\ComProbs\ca.run
```

Figure 11. The edit option.

3.2.6 Editing a file

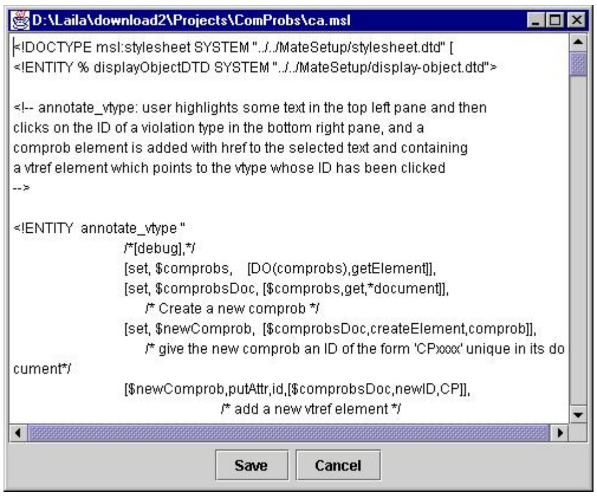
The MATE workbench uses three basic file types. Coding files are XML files. Stylesheets are MSL files used for visualisation of codings and for annotation support. Runfiles specify which stylesheet to apply to which XML file. Any XML, MSL or runfile in a MATE project folder can be opened and edited (cf. Figures 12-14) by selecting the file and clicking on 'Edit' in the projects window, cf. Figure 11.



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE comprobs SYSTEM "comprobs.dtd" [
<!ENTITY word_file "../MATE-dialogue/MATE-dialogue.xml">
<!ENTITY vtypes_file "vtypes_MATE-dialogue.xml">
]>

<!-- Trips test dialogue -->
<comprobs id="comprobs_MATE-dialogue">

<comprob id="CP1" href="&word_file;#id(utt_1)..id(utt_2)">
  <vtref id="VT1" href="&vtypes_file;#id(SG4-1)"/>
</comprob>

<comprob id="CP2" href="&word_file;#id(utt_1)..id(utt_2)">
  <vtref id="VT2" href="&vtypes_file;#id(SG5-1)"/>
</comprob>

<comprob id="CP3" href="&word_file;#id(utt_1)..id(utt_2)">
  <vtref id="VT3" href="&vtypes_file;#id(SG7-1)"/>
</comprob>
```

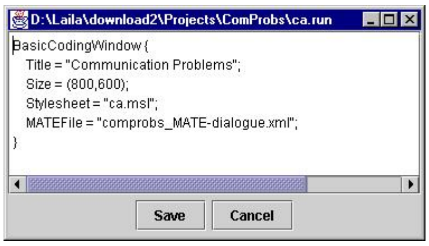
Figure 12. Editing an XML file.

.



```
<!DOCTYPE msl:stylesheet SYSTEM "../../MateSetup/stylesheet.dtd" [
<!ENTITY % displayObjectDTD SYSTEM "../../MateSetup/display-object.dtd">

<!-- annotate_vtype: user highlights some text in the top left pane and then
clicks on the ID of a violation type in the bottom right pane, and a
comprob element is added with href to the selected text and containing
a vtref element which points to the vtype whose ID has been clicked
-->

<!ENTITY  annotate_vtype "
                    /*[debug],*/
                    [set, $comprobs,    [DO(comprobs),getElement]],
                    [set, $comprobsDoc, [$comprobs,get,*document]],
                        /* Create a new comprob */
                    [set, $newComprob,  [$comprobsDoc,createElement,comprob]],
                        /* give the new comprob an ID of the form 'CPxxxx' unique in its do
cument*/
                    [$newComprob,putAttr,id,[$comprobsDoc,newID,CP]],
                                    /* add a new vtref element */
```

Figure 13. Editing a stylesheet.

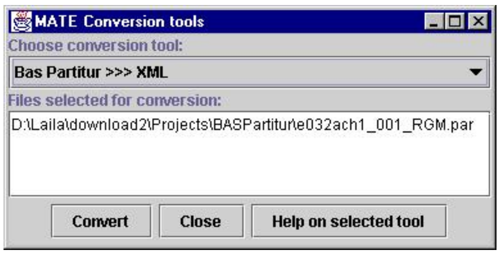


```
BasicCodingWindow {
   Title = "Communication Problems";
   Size = (800,600);
   Stylesheet = "ca.msl";
   MATEFile = "comprobs_MATE-dialogue.xml";
}
```

Figure 14. Editing a runfile.

3.2.7 Import from and export to other file formats

Files may be imported from other formats to XML. For the moment, conversion from XLabels and BAS Partitur to XML is enabled. New converters can easily be added, including converters which export from XML to other formats, such as HTML. Figure 15 shows import from BAS Partitur.



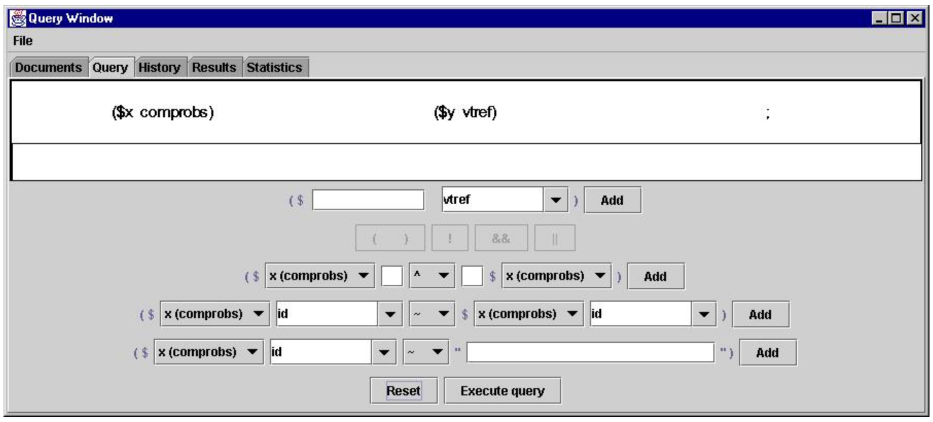
Choose conversion tool: Bas Partitur >>> XML

Files selected for conversion: D:\Laila\download2\Projects\BASPartitur\e032ach1_001_RGM.par

Figure 15. Import from BAS Partitur to XML.

3.3 Extracting information from annotated corpora

Once a corpus has been annotated, it must be possible to extract information from it for many different purposes. The MATE query tool is available for selecting the document(s) to be queried and for specifying the information to be subjected to a query, cf. Figure 16 (Isard et al. 2000). The query tool is activated from the tools menu in a coding window, cf. Figures 3 and 4. The information which can be extracted includes statistical information. Results are shown as illustrated in Figure 17. The interface for displaying results is not yet finalised. This is why the query results window for the moment only shows the raw XML data extracted from the queried XML file.



```
($x comprobs)        ($y vtref)        ;
```

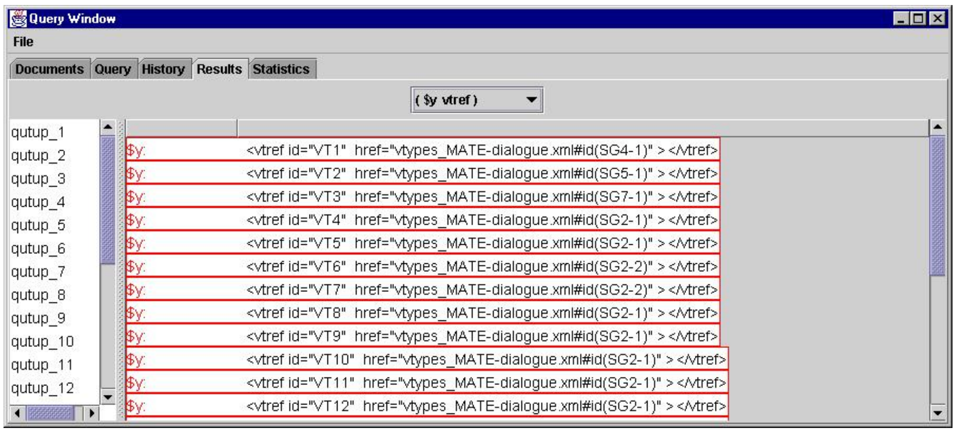
($ [ ] vtref) Add

($ x (comprobs) [ ] ^ [ ] $ x (comprobs)) Add

($ x (comprobs) id ~ $ x (comprobs) id) Add

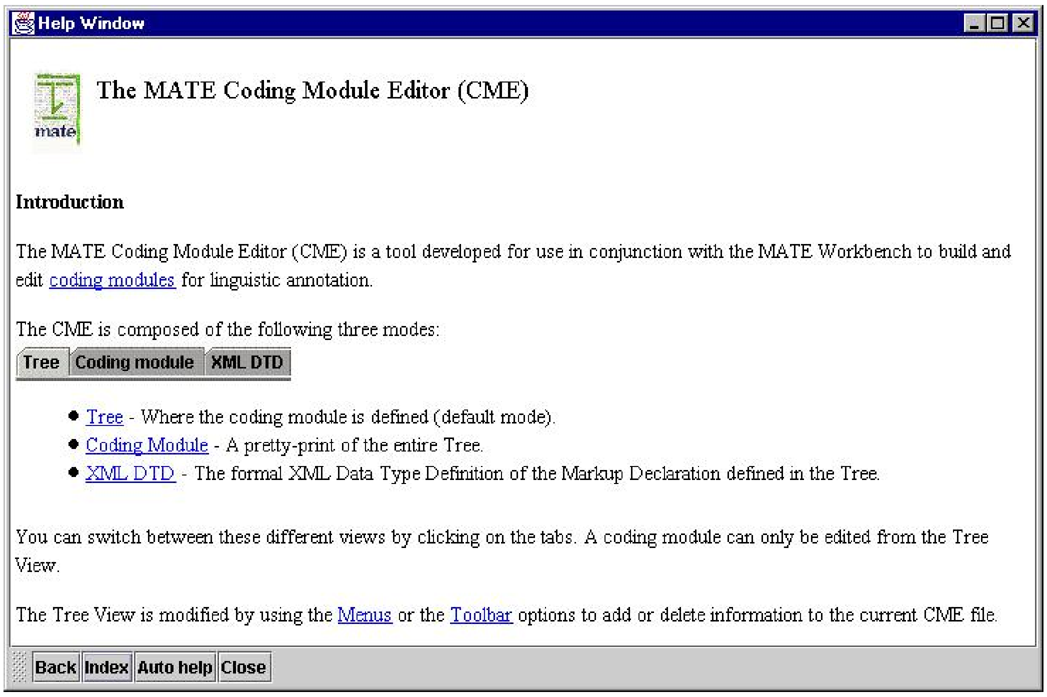
($ x (comprobs) id ~ " ") Add

Figure 16. The query window.



```
qutup_1    $y:   <vtref id="VT1"  href="vtypes_MATE-dialogue.xml#id(SG4-1)" > </vtref>
qutup_2    $y:   <vtref id="VT2"  href="vtypes_MATE-dialogue.xml#id(SG5-1)" > </vtref>
qutup_3    $y:   <vtref id="VT3"  href="vtypes_MATE-dialogue.xml#id(SG7-1)" > </vtref>
qutup_4    $y:   <vtref id="VT4"  href="vtypes_MATE-dialogue.xml#id(SG2-1)" > </vtref>
qutup_5    $y:   <vtref id="VT5"  href="vtypes_MATE-dialogue.xml#id(SG2-1)" > </vtref>
qutup_6    $y:   <vtref id="VT6"  href="vtypes_MATE-dialogue.xml#id(SG2-2)" > </vtref>
qutup_7    $y:   <vtref id="VT7"  href="vtypes_MATE-dialogue.xml#id(SG2-2)" > </vtref>
qutup_8    $y:   <vtref id="VT8"  href="vtypes_MATE-dialogue.xml#id(SG2-1)" > </vtref>
qutup_9    $y:   <vtref id="VT9"  href="vtypes_MATE-dialogue.xml#id(SG2-1)" > </vtref>
qutup_10   $y:   <vtref id="VT10"  href="vtypes_MATE-dialogue.xml#id(SG2-1)" > </vtref>
qutup_11   $y:   <vtref id="VT11"  href="vtypes_MATE-dialogue.xml#id(SG2-1)" > </vtref>
qutup_12   $y:   <vtref id="VT12"  href="vtypes_MATE-dialogue.xml#id(SG2-1)" > </vtref>
```

Figure 17. Results of a violation types query, cf. Figure 4.



The MATE Coding Module Editor (CME)

Introduction

The MATE Coding Module Editor (CME) is a tool developed for use in conjunction with the MATE Workbench to build and edit coding modules for linguistic annotation.

The CME is composed of the following three modes:

Tree | Coding module | XML DTD

- Tree - Where the coding module is defined (default mode).
- Coding Module - A pretty-print of the entire Tree.
- XML DTD - The formal XML Data Type Definition of the Markup Declaration defined in the Tree.

You can switch between these different views by clicking on the tabs. A coding module can only be edited from the Tree View.

The Tree View is modified by using the Menus or the Toolbar options to add or delete information to the current CME file.

Figure 18. The help window.

3.4 Getting help

An online help facility may be consulted at any time during use of the MATE workbench. Figure 18 shows the topmost help page for the coding module editor. Explanation of particular coding schemes is available from the help menu in the coding window, cf. Figures 3 and 4.

3.5 Usability

Usability is a key concern in MATE. The focus on usability is reflected in the coding module concept and in the workbench coding module editor. The coding module prescribes a comprehensible standard description to be made of any coding scheme. The coding module editor makes it easy to specify coding modules – not least their markup declarations - also for non XML-literate users. The editor enables the user to specify the markup declaration for a new coding module almost without requiring any knowledge of the underlying XML representation. The coding module editor automatically generates a DTD which is then used internally by the workbench. The coding module editor thus represents a major step forward compared to tools which require users to write DTDs.

There are still a couple of major usability issues to be solved, however. One issue is a user-friendly way of creating new coding visualisations. Writing style sheets for the workbench is cumbersome and requires programming skills because no editor is provided. The user must edit the raw style sheet code (or write new code), cf. Figure 13. It is high on our wishlist to enable users to easily define new visualisations. This may be done either by providing a stylesheet editor comparable to the coding module editor as regards ease of use, or, alternatively, through a completely new interface concept replacing the need for stylesheets and enabling users to easily define new visualisations.

A second major issue is the interface to the query tool and its results. As for the latter, Figure 17 makes it evident that usability improvements are needed. The query results could be presented far more transparently using an appropriate style sheet. This is also on the MATE to-do list. So is a more comprehensible interface for expressing queries than the present one (Figure 16).

4. STATE OF THE ART

Several frameworks for speech corpus annotation have been proposed but to our knowledge the MATE markup framework is still the more comprehensive framework around. An example of another framework is the annotation framework recently proposed by Bird and Liberman (1999) which is based on annotation graphs. These are now being used in the ATLAS project (Bird et al. 2000) and in the Transcriber tool (Geoffrois et al. 2000). The annotation graphs serve as an intermediate representation layer between interface and internal data structures. Whilst Bird and Liberman do not consider coding modules or discuss the interface from a usability point of view, they present detailed considerations concerning time line representation and time line reference. The two frameworks may, indeed, turn out to complement each other nicely.

Acknowledgements

We gratefully acknowledge the support for the MATE project provided by the European Commission’s Telematics/Language Engineering Programme. We would also like to thank all MATE partners. Without the very considerable joint efforts of the project consortium it would not have been possible to build the MATE workbench.

5. REFERENCES

The MATE workbench is available in executable version and under the GNU open source LGP license from the MATE web site at http://mate.nis.sdu.dk. MATE reports are also available from this web site.

Bird, S. and Liberman, M.: *A Formal Framework for Linguistic Annotation.* Technical Report MS-CIS-99-01. Department of Computer and Information Science, University of Pennsylvania, 1999.

Bird, S., Day, D., Garofolo, J., Henderson, J., Laprun, C. and Liberman, M.: ATLAS: A Flexible and Extensible Architecture for Linguistic Annotation. *Proceedings of the 2nd International Conference on Language Resources and Evaluation* (LREC 2000), Athens, 2000, 1699-1706.

Dybkjær, L. and Bernsen, N. O.: The MATE Workbench. *Proceedings of the LREC’2000 workshop on Data Architectures and Software Support for Large Corpora,* Athens, 2000, 33-37 (a).

Dybkjær, L. and Bernsen, N. O.: The MATE Markup Framework. *Proceedings of the 1st SIGdial Workshop on Discourse and Dialogue,* Hong Kong, 2000 (b).

Dybkjær, L., Bernsen, N. O., Dybkjær, H., McKelvie, D. and Mengel, A.: *The MATE Markup Framework.* MATE Deliverable D1.2, 1998.

Geoffrois, E., Barras, C., Bird, S. and Wu, Z.: Transcribing with Annotation Graphs. *Proceedings of the 2nd International Conference on Language Resources and Evaluation* (LREC 2000), Athens, 2000, 1517-1521.

Isard, A., McKelvie, D., Cappelli, B., Dybkjær, L., Evert,S., Fitschen, A., Heid, U., Kipp, M., Klein, M., Mengel, A., Møller, M. B. and Reithinger, N.: *Specification of Work-bench Architecture.* MATE Deliverable D3.1, 1998.

Isard, A., McKelvie, D., Mengel, A., Møller, M. B., Grosse, M. and Olsen, M. V.: *Data Structures and APIs for the MATE Workbench.* MATE Deliverable D3.2, 2000.

Klein, M., Bernsen, N. O., Davies, S., Dybkjær, L., Garrido, J., Kasch, H., Mengel, A., Pirrelli, V., Poesio, M., Quazza, S. and Soria, C.: *Supported Coding Schemes.* MATE Deliverable D1.1, 1998.

Mengel, A., Dybkjær, L., Garrido, J., Heid, U., Klein, M., Pirrelli, V., Poesio, M., Quazza, S., Schiffrin, A. and Soria, C.: *MATE Dialogue Annotation Guidelines.* MATE Deliverable D2.1, 2000.