Project ref. no.	IST-1999-10647
Project title	ISLE Natural Interactivity and Multimodality Working Group

Deliverable status	Public
Contractual date of delivery	October 2000
Actual date of delivery	January 2001
Deliverable number	D11.1
Deliverable title	Survey of Existing Tools, Standards and User Needs for Annotation of Natural Interaction and Multimodal Data.
Type	Report
Status & version	Final
Number of pages	101
WP contributing to the deliverable	WP11
WP / Task responsible	Laila Dybkjær, NISLab
Author(s)	Laila Dybkjær, Stephen Berman, Michael Kipp, Malene Wegener Olsen, Vito Pirrelli, Norbert Reithinger, Claudia Soria
EC Project Officer	Brian Macklin
Keywords	Tools, standards, user needs, annotation, natural interaction and multimodal data.
Abstract (for dissemination)	This ISLE Deliverable 11.1 from the ISLE Natural Interactivity and Multimodality Working Group presents a survey of some of the most prominent tools in support of natural interactivity and multimodal data annotation, i.e. tools which support annotation of spoken dialogue, gesture, facial expression, or cross-modality issues. The report reviews 12 different tools. A couple of these have not been implemented yet. However, the tool concepts involved were found sufficiently interesting for inclusion in this report, e.g. because they have standardisation as a goal. The tools descriptions use a common structure agreed upon early in the writing process. The purpose of this structure is to facilitate comparison across tools by providing similar information about each tool to the extent possible. The final section of the report compares the reviewed tools and discusses user needs and standardisation efforts.

ISLE Natural Interactivity and Multimodality Working Group Deliverable D11.1

Survey of Existing Tools, Standards and User Needs for Annotation of Natural Interaction and Multimodal Data

January 2001

Authors

Laila Dybkjær[1], Stephen Berman[2], Michael Kipp[4], Malene Wegener Olsen[1], Vito Pirrelli[3], Norbert Reithinger[4], Claudia Soria[3]

1: NISLab, Odense University, Denmark. 2: IMS, Stuttgart University, Germany.

3: ILC, Pisa, Italy. 4: DFKI, Saarbrücken, Germany.

Section responsibilities

Section 1:	Introduction	Laila Dybkjær
Section 2:	Anvil	Michael Kipp
Section 3:	ATLAS	Malene Wegener Olsen and Laila Dybkjær
Section 4:	CLAN	Vito Pirrellí
Section 5:	CSLU Toolkit	Stephen Berman
Section 6:	The MATE Workbench	Laila Dybkjær
Section 7:	MPI Linguistic Tools: CAVA and EUDICO	Stephen Berman
Section 8:	Multitool	Malene Wegener Olsen and Laila Dybkjær
Section 9:	The Observer	Vito Pirrelli
Section 10:	SignStream	Claudia Soria
Section 11:	SmartKom	Norbert Reithinger
Section 12:	syncWRITER	Claudia Soria
Section 13:	TalkBank	Malene Wegener Olsen and Laila Dybkjær
Section 14:	User needs and standards	Laila Dybkjær

Contents

1 Introduction

The aim of this ISLE Natural Interactivity and Multimodality Working Group report is to provide a survey of some of the most prominent tools in support of natural interactivity and multimodal data annotation, i.e. tools which support annotation of spoken dialogue, facial expression, gesture, bodily posture, or cross-modality issues.

The report reviews twelve different tools (Sections 2-13). A couple of these have not been implemented yet. However, the tool concepts involved were found sufficiently interesting for inclusion in this report, e.g. because they have standardisation as a goal.

The tools and projects which have been reviewed are the ones listed below. A '+' after the tool/project name indicates that the section has been written by the tool developer or by a person deeply involved in the project. A '*' indicates that the section was written by a person external to the tool/project but that one or more people working on the tool/project were invited to read and comment on the section and that their feedback has been taken into account.

- *Anvil*[+] (Annotation of Video and Language Data) is a Java-based annotation tool for video files.
- *ATLAS*[*] (Architecture and Tools for Linguistic Analysis Systems). No tool is available yet.
- *CLAN*[*] (Computerized Language Analysis) is a program designed specifically to analyse data transcribed in the format of the Child Language Data Exchange System (CHILDES). Transcripts can be linked to audio or video files.
- *CSLU Toolkit*[*] (Center for Spoken Language Understanding Toolkit) is a suite of tools including the Rapid Application Developer, BaldiSync (for facial animation), SpeechView, OGIsable (an annotation tool), speech recognition tools, and a programming environment (CSLUsh). OGIsable is the only annotation tool included in the CSLU Toolkit. It allows the user to attach properties to a text before it is spoken (via the Festival TTS engine), e.g. to synthesize facial expression synchronised with speech output.
- *MATE*[+] (Multilevel Annotation Tools Engineering). The MATE Workbench is a Java-based tool in support of multilevel annotation of spoken dialogue corpora and information extraction from annotated corpora.
- *MPI tools*[*] (CAVA and EUDICO) (Computer Assisted Video Analysis and European Distributed Corpora). Both are tools in support of annotation of audiovisual files and information extraction. Eudico is still at an early stage.
- *MultiTool*[*] (developed as part of a Swedish project on a Platform for Multimodal Spoken Language Corpora). This is a Java-based tool in support of the establishment and use of multimodal spoken language corpora (audio and video).
- *The Observer*[*] is a professional system for the collection, analysis, presentation and management of observational data. It can be used to record activities, postures, movements, positions, facial expressions, social interactions or any other aspect of human or animal behaviour as time series of tagged data.
- *Signstream*[*] (developed as part of the American Sign Language Linguistic Research Project) is a database tool for analysis of linguistic data captured on video.
- *SmartKom*[+] is a large-scale German project at the specification stage which aims to merge the advantages of dialogue-based communication with the advantages of a mixture of graphical user interfaces and gesture and mimic interaction. SmartKom uses tools developed in the Verbmobil project for audio annotation. For annotation of mimics and gestures the Anvil tool (see Section 2) and a commercial product are being considered.
- *SyncWriter*[*] is a transcription and annotation tool designed to provide help for transcription and annotation of synchronous "events" such as speech and video data.
- *TalkBank*[*] is a US project which aims to provide standards and tools for creating, searching, and publishing primary materials via networked computers. No tools have been developed by Talkbank yet. At this point, a couple of tools are included which were made as part of other projects (CLAN, Transcriber).

The twelve tool reviews are alphabetically ordered in the report. The tool descriptions use a common structure agreed upon early in the writing process. The purpose of this structure is to facilitate comparison across tools by providing similar information about each tool to the extent possible. The common structure has seven main entries as shown in Figure 1.1. Entries may be merged or left out in the individual reviews depending on the information available and how it was best presented. Under each main entry, a set of more specific information items (the bulletted items in Figure 1.1) have been added to serve as guidelines for which information it would be desirable to include under each entry.

Introduction
- Aim of the tool
- Which domain does the tool cover
- Which task(s) does is solve
- What was the reason for creating the tool (user needs, own needs, curiosity, ...)
- Which version of the tool is being reviewed
- Is a demo available
- If yes, is the demo freely available or is there a fee
- Is the review based on hands-on experience with the tool, on written descriptions, or on which other sources (include them in the reference list)

Software design
- Architecture
- Implementation language(s)
- Other software related issues

Platform requirements
- Operating system(s) on which the tool runs
- Special hardware requirements, if any

Interface
- Description of interface design and usability of the tool
- Include screen shots if possible
- Advantages and disadvantages of the interface

Functionality
- Description of each main functionality plus example(s) for each
- Does the tool offer the functionality it promises
- How useful is it
- Advantages and disadvantages

Conclusion
- General assessment of usability and functionality of the tool
- How interesting is the tool (whole tool, particular aspects) for our long-term purpose of creating a tool in support of annotation of natural interactivity and multimodal data

References

Figure 1.1. The common structure used for tool reviews.

Several tools claim to provide general natural interactivity and multimodality annotation support and/or to work towards some kind of standard while others have rather been built to solve a particular task in a specific project. Section 14 compares the reviewed tools and outlines standardisation efforts and user needs reflected in the tools and the projects which created them.

2 Anvil - Annotation of Video and Language Data

2.1 Introduction

Anvil is a Java-based annotation tool for video files[1]. It allows the encoding of nonverbal behaviour (e.g. gesture) and import of annotations of speech related phenomena (e.g. dialogue acts) on multiple layers, so-called tracks[2] and their visualization in temporal alignment. Structuring the tracks and defining their respective properties according to a specific annotation scheme can be done with Anvil's generic track configuration. Data storage and exchange is all handled in XML.

Certain annotations must be handled by other, external programs ("Praat" for speech labelling, "RST Tool" for coding of rhetorical structure, both public domain) and subsequently imported into Anvil which translates them to corresponding tracks. Statistical analysis of the data is not supported but can be performed by working on the XML output files.

Anvil was written as part of an ongoing PhD project on nonverbal communication sponsored by the DFG (Deutsche Forschungsgemeinschaft) and conducted at DFKI and the University of the Saarland. It has been used to encode video samples of a German TV show with nonverbal communication events and linguistic information. In software and interface design Anvil builds on experiences with mass corpus annotation (dialogue acts) made within the VERBMOBIL II project.

The system is freely available for research purposes from http://www.dfki.de/~kipp/anvil. The current release version is 2.2.

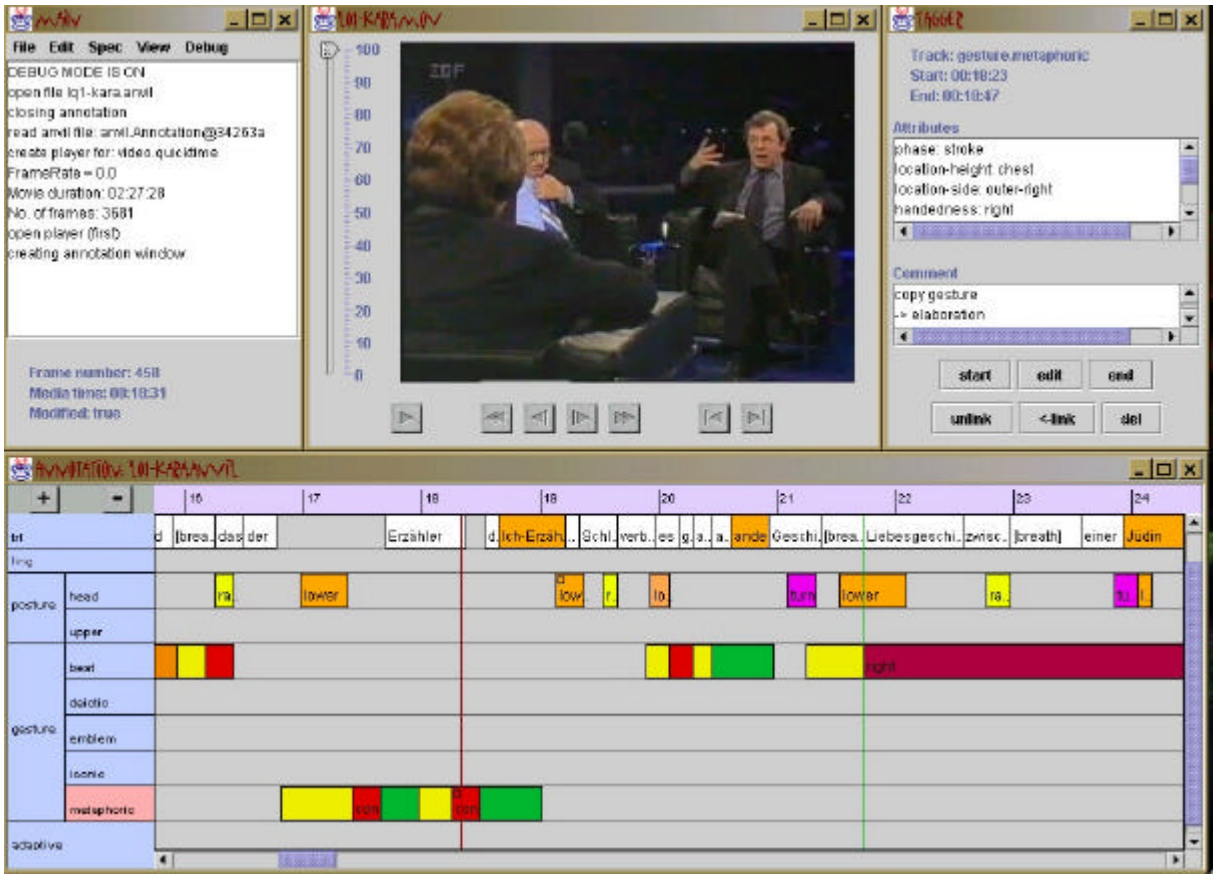


Figure 2.1. All windows of the Anvil system.

[1] Currently, the only supported video format is QuickTime. This is due to the Java Media Framework (JMF, version 2.0) where frame-wise control is only possible with QuickTime files.

[2] Also called tiers, layers or levels in other applications.

Figure 2.1 gives an impression of the overall system. Almost in the centre is the *video display* with the usual VCR controls. Below, taking up the lower half of the screen, is the *annotation board*, the essential work space for annotation: Tracks are shown as horizontal bars where elements for words, movements etc. can be added and edited with the help of the *track window* (upper right). The *main window* (upper left) holds the main menu bar for file selection, view options etc. as well as a text display for trace/debug information.

2.2 Software Design

This section presents Anvil's annotation framework which is designed to accommodate a number of possible schemes. A closely related topic, file organization, will be treated at the end of this section.

Anvil's overall design is object-oriented, the largest unit being an *annotation*. This annotation object is the container for all the information you want to encode for one *video file* (or a number of videos when you shot the same scene/session from different angles). The annotation depends on a *specification file* which is a formal description of the annotation scheme. In Anvil this scheme must be formulated in terms of hierarchical tracks and track elements that can hold a number of attributes.

2.2.1 Tracks

A track is a container for one type of information. Examples are: words (transliteration), deictic gestures, head nods. Each track can contain a number of *track elements*, e.g. a single word or a single head nod. These elements are viewed as objects that can not only be labelled but hold a number of attribute-value pairs. Thus, a word in the transliteration track could have attributes like "emphasis" or "loudness". Each attribute expects values of a certain *type* which can be a *set of labels* or the *String* type. E.g., the attribute "emphasis" has a set of possible labels: "strong", "moderate" and "reduced". On top of these attribute-value pairs, an arbitrary number of comment lines can be added to each track element for spontaneous, non-schematic descriptions.

There are three different types of tracks. What the example tracks above had in common is that they contain track elements with a start and end time relating to the video being analysed. This type of track is called a *primary track*. The other two types are secondary types because their elements relate to elements of another track (called the reference track). So-called singleton tracks have elements that point to exactly one partner element in the reference track. If the primary track "transliteration" contains words, a *singleton* track could store the words' corresponding part-of-speech label. Each element in the singleton track would point to a specific element in the primary track, inheriting the primary track element's start and end time.

The *span* track type allows an element to cover a number of contiguous elements in the reference track. This is useful to mark e.g. dialogue acts or rhetorical relations (although, for the latter, a hierarchical encoding of relations is not feasible).

2.2.2 Track hierarchy

Tracks can be grouped together by a *group node* which is equivalent to a track but cannot hold any elements and is therefore only a structural entity. You can use this for didactic or ergonomic reasons, e.g. to group linguistic tracks together (part-of-speech, rhetorical relations, dialogue acts) or all gesture-related tracks (beat, emblematic and illustrative gestures). Track and group names are constructed like path names in Java. Thus, the full name of the track "deictic" in the group "gesture" is "gesture.deictic".

Groups can have attributes and corresponding value sets which are *inherited* by the sub-tracks (and sub-groups). This can save some specification/documentation work when devising many tracks that share the same attributes.

```
<?xml version="1.0" ?>
<annotation-spec>
<head>
...
</head>

<body>
<track-spec name="transliteration" type="primary">
<attribute name="word" />
<attribute name="emphasis">
<value-el> strong </value-el>
<value-el> moderate </value-el>
<value-el> reduced </value-el>
</attribute>
</track-spec>
...

</body>
</annotation-spec>
```

Figure 2.2. Excerpt of a specification file.

2.2.3 Specification

You define all the tracks, groups, attributes and value types in the specification file. The specification file is an XML file and very straightforward to write. It contains the following information:

Tracks/Groups: structure of track hierarchy, plus track/group names and track types (primary, singleton, span)

Track elements: attribute names and corresponding value type (set of labels or String)

Display options: in the graphical interface (see Section 2.3), which attribute value to use as a label for a track element, which attribute value to use for colour-coding (specify a colour for each value)

Documentation: descriptions and definitions for tracks, elements, attributes and value labels for automatic manual generation (see Section 2.5.2)

Figure 2.2 shows an excerpt of a specification file where you see the track named "transliteration" defined, containing the attributes "word" and "emphasis".

2.2.4 File Organization

Anvil's file organization is centred around the annotation file (extension "anvil") which contains the data encoded for one video (or more videos showing the same session from different angles) and is based on a scheme defined in a specification file (extension "xml"). The paths of all video files as well as the specification file are contained in the annotation file (see Figure 2.3). There is no project file. Data like coder name, date of encoding etc. can be kept in the annotation file as well (not supported yet).

Other relevant files are the HTML files created by Anvil's coding manual generator (see Section 2.5.2) and those files that can be imported into the current annotation: TextGrid files produced by Praat will be fused into the transliteration track, rs2 files produced by the RST Tool are inserted in the RST track.

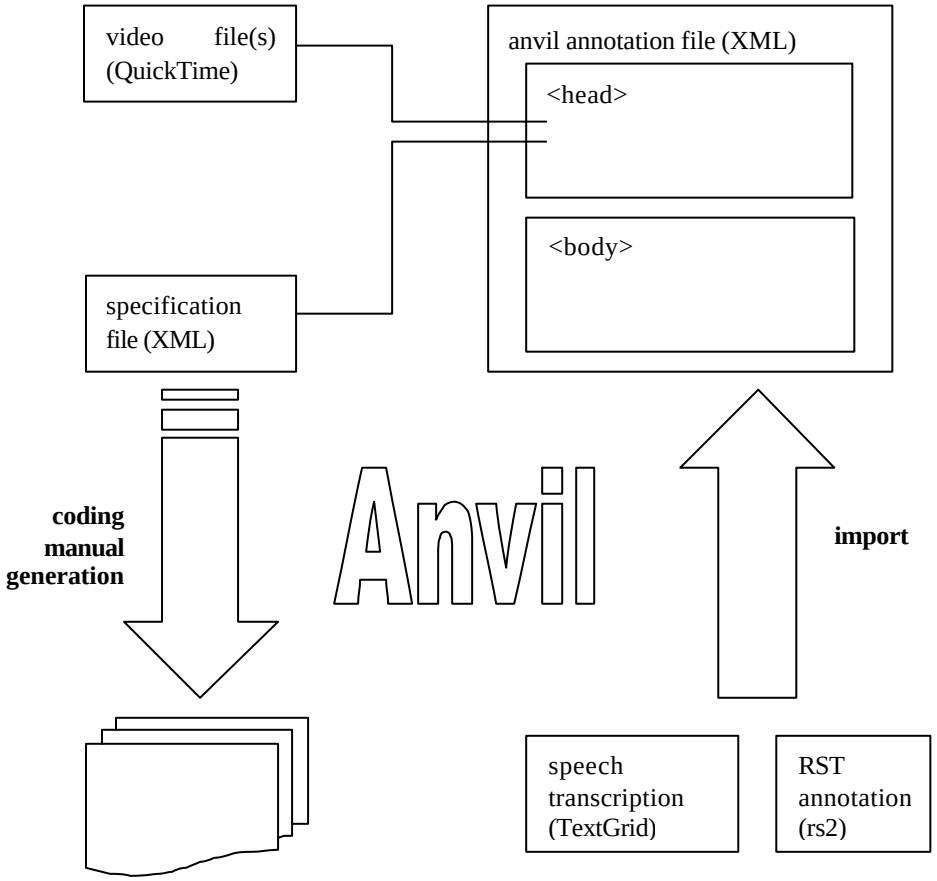


Figure 2.3. File organization in Anvil.

2.3 Platform Requirements

The system was developed on a Pentium III (500 MHz) with 128 MB RAM, and successfully tested on Windows98, Windows NT 4.0 and Windows2000 platforms. Since it is written in Java2 with JMF 2.0 (Java Media Framework) and xml4j[3] (XML parser by IBM Alphaworks) it should run on any platform that has those components installed.

2.4 Interface

This section will explain Anvil's graphical user interface and how its components are used in annotation. Each subsection treats one window (see Figure 2.1) in depth.

2.4.1 Main window

Anvil starts with its main window which holds the main menu bar and a text display for listing user actions and debugging information (see Figure 2.4). A hotlist in the File menu allows you to open the four most recently used files instantly without clicking through a file select menu.

To start an annotation the user can either open a video file (QuickTime format) or open a previously created annotation file (with "anvil" extension). In the latter case the corresponding video will

[3] Available from http://www.alphaworks.ibm.com/tech/xml4j.

automatically be loaded. In both cases, Anvil will present its four windows as shown in Figure 2.1: main, video, track and annotation window (from left to right, top to bottom).

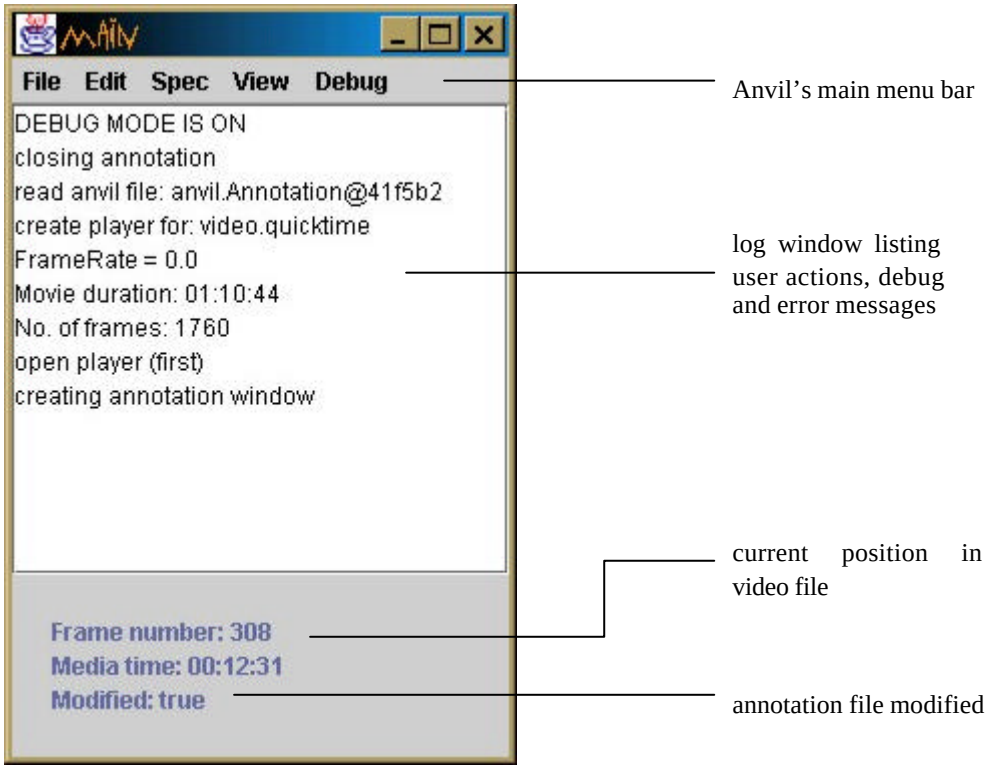


Figure 2.4. Main window.

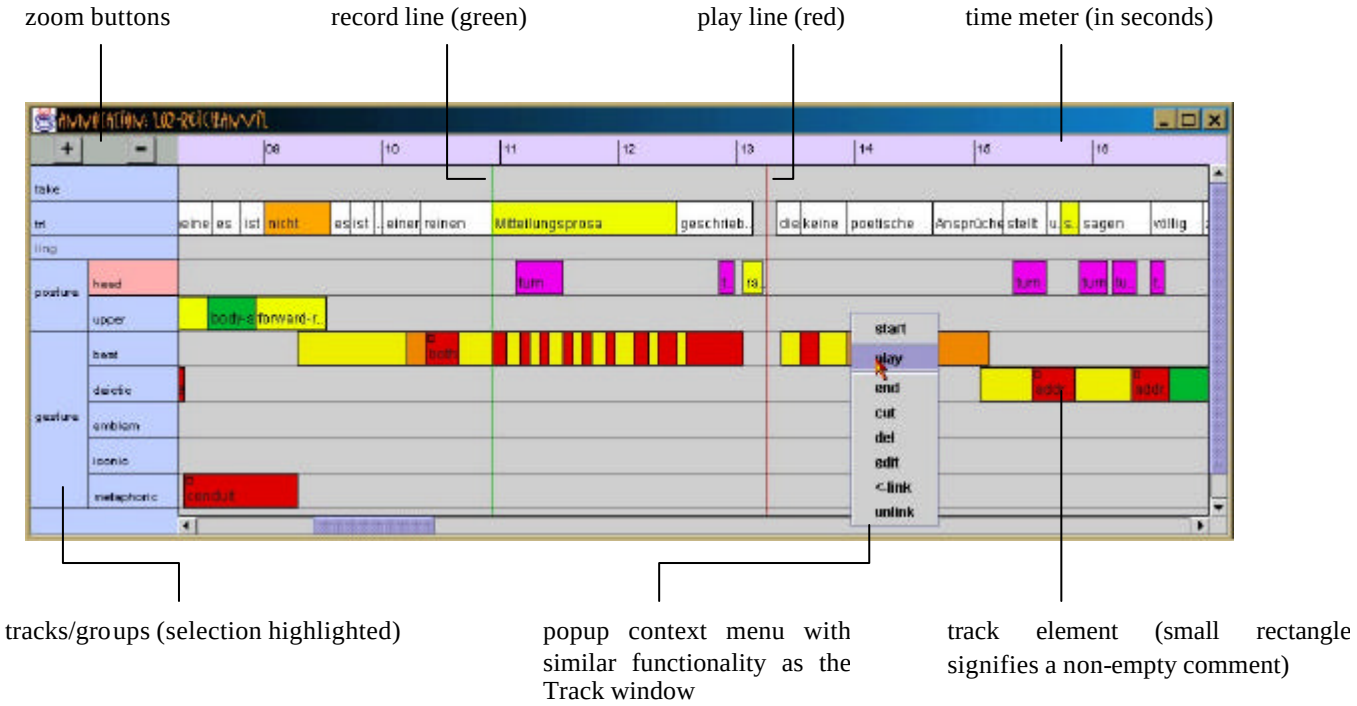


Figure 2.5. Annotation window.

2.4.2 Video window

The video window (top middle window in Figure 2.1) displays the video, providing the usual VCR controls for play/pause, frame-wise forward/backward stepping and jumping to beginning and end. Along the left side there is a speed slider for slow motion playback.

2.4.3 Annotation board

This window (Figure 2.5) gives a time aligned view on the tracks and its contained elements. The track hierarchy is seen on the left, the active track being highlighted. Tracks are ordered as they appear in the specification file (see Section 2.2.3). The larger section on the right shows the track elements as boxes. Time, represented by the x-axis, is marked in seconds on the top bar. The top left corner provides zoom in/out buttons where zoom factor one represents a ratio of one pixel per frame (usually 1/25 seconds). A red vertical line, the *play line*, marks the current position in the video. By dragging the line the user can navigate through the video. Clicking on the annotation board causes the play line to be positioned on the click point and the respective track to be selected.

The *track elements* are labelled with the value of their most important attribute as specified by the user (see Section 2.2.3). E.g., in the transliteration track (abbreviated "trl") the attribute "word" is displayed. A second attribute can be displayed by means of colour-coding. Each value can be assigned a specific colour that is used to fill the track elements' box.

In order to insert a new track element the user must mark begin and end time by placing the green line at the beginning, the red line at the end of the element. The red *play line* always follows the video's frame position in playback. The record line does not move but can be placed at the current position of the red line by pressing the "start" button in the popup context menu (right mouse button), also reachable through the Track window (see Section 2.4.4 below). The user must then place the red line at the end of the element and press "end" to trigger an edit menu for the data input (see Section 2.4.5). The "play" option in the context menu allows the user to play back the portion of the video between start and play line.

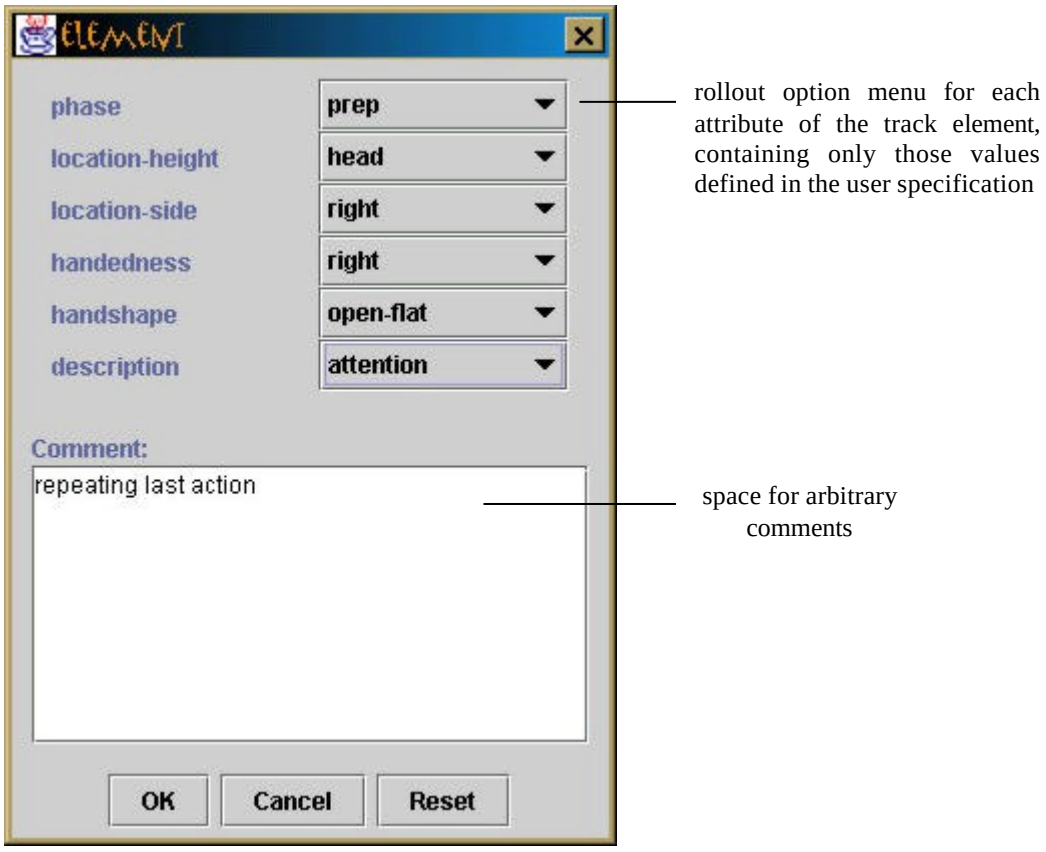


Figure 2.6. Track window.

2.4.4 Track window

In this window the system gives a detailed view on the currently selected track element (Figure 2.6). The "currently selected track element" is that element in the selected track (highlighted pink, see Figure 2.5) which is intersected by the play line (red). Its content, i.e. all attributes and values (upper text area) plus the comment lines (lower text area) can be observed here. On the very top there is the full name of the currently selected track (e.g. "gesture.deictic"), followed by start and end time of the currently selected track element. This window also allows the addition of new elements by clicking the "end" button (with the play line at the desired end time) which will trigger the element edit window (see Section 2.4.5 below). Other functions are the deletion and editing of existing elements and the *linking* of subsequent elements to element groups.

2.4.5 Element edit window

The edit window pops up when the user wants to create a new or edit an existing track element (Figure 2.7). It shows all the attributes applicable for the current track and offers the selection of legal value labels for each of them or a text input area for String values. The lower text area expects textual input for any comments concerning this element.

2.5 Functionality

The tool's main function is coding on different levels or tracks. The steps necessary for successful annotation are listed in the next section. A side feature is automatic manual generation which is explained in the second section. To give an impression of the actual scheme that Anvil was developed to work with, the last section elaborates aspects of the project that led to the development of Anvil.

2.5.1 Coding

The first requirement for annotation is the digitisation of the video material. It is prerequisite that its final format is QuickTime using one of the codecs[4] supported by JMF[5]. Once the data material is secured you need to design an annotation scheme, deciding which behavioural elements you want to encode and how. Now, the technical part can begin:

Specification: From your annotation scheme, create a specification of the tracks (possibly a hierarchy using group nodes), plus the attributes and values types for the track elements.

Data Import

Transcription: For most projects it is necessary to have a full transcription of the speech occurring in the video. Use the (public domain) program *Praat* to do it. With its function *speech labelling* create one layer which you call "words" and where you insert all the transcribed words. Write the resulting data to a "short TextGrid" file. Import the TextGrid file into Anvil.

Rhetorical Structure: If rhetorical structure is relevant for the project use the *RST Tool* by Mick O'Donnell for coding. This tool needs ASCII text input that can be generated by Anvil. The RST annotated text can be imported back to Anvil.

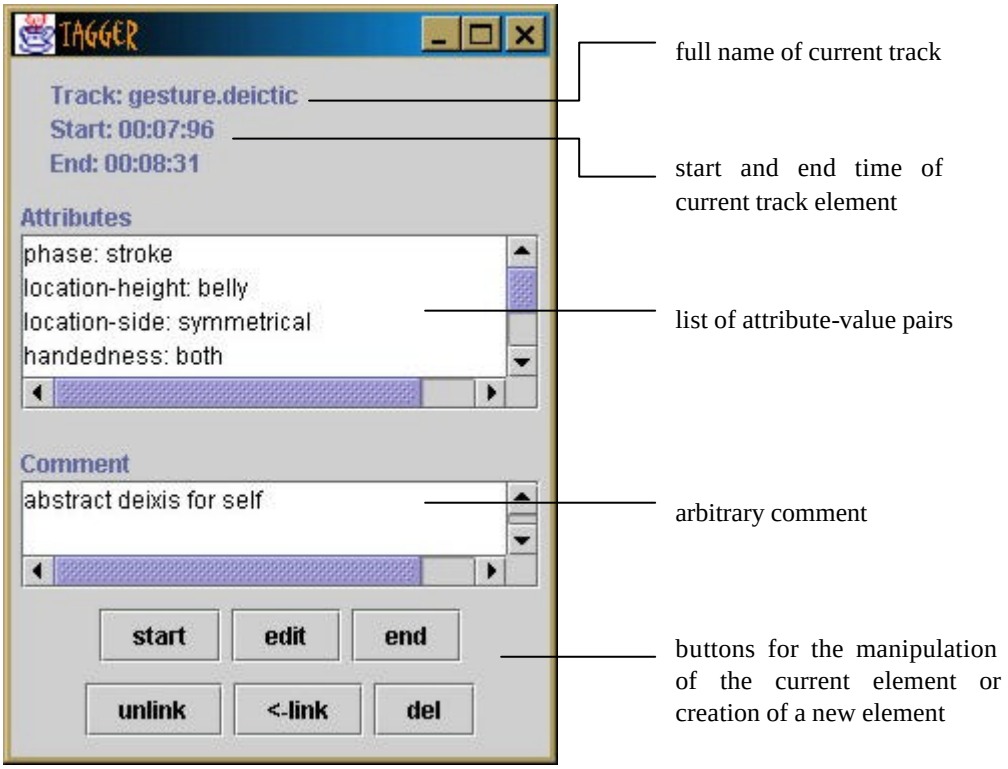
Annotation: By positioning the play line at start and end positions of identified nonverbal events, mark out and edit the track elements you need to capture. In many cases it is advisable to code one track or one group at a time for the whole length of a video. Write the resulting annotation to an XML Anvil file.

The resulting XML files can be read into analysis components for statistical verification of hypotheses and for inter-coder and intra-coder reliability studies. This is not provided by Anvil.

---

[4] Whereas the term QuickTime refers to a media architecture, the actual **co**ding and **dec**oding of video data is done by the so-called codec. The QuickTime specification allows arbitrary codecs to be integrated. A specific piece of software, however, like JMF, only supports a limited range of codecs.

[5] For JMF version 2.0 supported video codecs are: Cinepak, H.261, H.263, Indeo (iv31 and iv32), JPEG (411, 422, 111) and RGB. For an up-to-date listing visit: http://java.sun.com/products/java-media/jmf/2.1/formats.html



2.7. Element edit window.

2.5.2 Coding Manual Generation

For the systematic study of behaviour it is crucial that the identified units are defined as clearly and consistently as possible. Putting such definitions in words and writing an accompanying document, the *coding manual*, can be painstaking work. Anvil does not spare you all the work but alleviates the task by exploiting the fact that the structure of the coding manual is usually the same as the one already defined in the *specification file*. By inserting definitions and descriptions into the XML specification file (bracketed by <doc> tags) you can have an HTML manual generated which has the look-and-feel of javadoc (see Figure 2.8). The advantage of HTML being that you can browse the electronic manual on screen while in the process of annotation.

2.5.3 A Sample Project

The PhD project (work in progress, no publications yet) that led to the development of Anvil works on the crossroads of psychology and embodied agents research in an effort to make agent behaviour more "life-like". Much work has been done in psychology, semiotics and anthropology to understand and categorize human nonverbal behaviour, i.e. gesture, facial expression, posture etc. Some of the knowledge found in this research is already actively being used by computer scientists and animation artists (see e.g. Cassell et al., 2000). This interdisciplinary exchange which was very much focused on facial expression is now moving to more general theories of nonverbal communication. Whereas some *general* findings have become established, comparatively little has been done to try to find rules of individual behaviour and apply these for animated agents in a systematic way. The project described here aims at the latter: trying to develop a methodology starting from an analysis of individual behaviour (using the Anvil system) and bridging the gap to computer animation with a rule-based generator that produces "life-like" gesture and posture from an input of "structured" speech.

The employed annotation scheme falls into two categories: linguistic and body motion events. Linguistic entities are:

Emphasis: emphasis of single words, encoded in four steps (using the SABLE tags): reduced, none, moderate, strong.

Intonation groups: ranges of words within the bounds of one intonation curve. Annotated by a trained phonetician using a very fine-grained segmentation. Since intonation phrases can be identified on different levels of granularity, annotation is somewhat subjective.

Rhetorical relations: groups of words are identified as segments according to RST by Mann/Thompson and annotated using an extended set of their relations. The annotation is imported from the RST Tool (version 2.5) and the hierarchy flattened. Each segment has a relation name and a direction (which way the arrow of the relation is pointing: forward or backward).

The body events fall into two groups: posture and gesture. The posture subcategories relate to body parts (currently: head and upper body), whereas the gesture categories follow a functional approach quite common in psychology (cf. McNeill92):

Beat gestures are rhythmical movements accompanying words and phrases without bearing any visual relationship to the contents.

Deictic gestures are pointing gestures marking either a present object/person or an abstract entity like in "it was well prepared and [that] was good" where "that" is accompanied by a pointing gesture which refers to "it was well prepared".

Emblematic gestures follow a culture-specific code to signify a message that could also be expressed in words, e.g. the thumbs-up gesture which could be replaced by "OK" or "good".

Iconic gestures bear some relationship to the content of the accompanied speech, e.g. drawing a round shape in the air when talking about a ball or making a writing gesture when asking for a signature.

Metaphoric gestures are similar to iconic gestures, only that the metaphoric gesture relates to an object or activity that stands metaphorically for the speech content. E.g. indicating a block with your hand where that blocks represents a house or a bank account.

One single track was assigned to each of these categories instead of using only one "gesture" track for two reasons: First, it is possible that the same gesture falls in more than one category or that two gestures of different categories overlap. Second, each category requires its own distinct set of element properties, i.e. attribute-value pairs.

The attributes of the gesture tracks (some apply to all tracks and can therefore be specified in the group node "gesture", some are track-specific) cover different aspects of the identified event. Here are some examples:

phase: a structural property, distinguishing between preparation, stroke, retraction and partial retraction (cf. McNeill, 1992, and Kita et al., 1998).

location: the height and lateral position in gesture space.

form: the shape or activity indicated by an iconic.

semantics: the conventionalized meaning of the emblem.

See Section 2.9 (appendix) for an excerpt of an example annotation.

2.6 Related Work

In linguistic and multi-modal data annotation one has to distinguish between (1) annotation schemes, (2) annotation frameworks[6] and (3) tools. Tools, again, can handle different sorts of tasks: (a) annotation/coding, (b) data retrieval and (c) analysis. Not all projects cover all aspects. I will review very briefly those projects that are related to Anvil and point out some of the differences.

The Partitur format (Schiel et al. 1998) developed by the Bavarian Archive for Speech Data (BAS) was used in the VERBMOBIL project (Wahlster 2000) and is an annotation framework (2) that supports multiple tracks. There is one reference track which all other tracks refer to, the so-called canonical track (containing words in a phonological transcription). The elements of other tracks can point to an arbitrary collection of elements of the canonical track. The description of an element is

[6] I take an annotation framework to be a representation mechanism for the entities of an annotation scheme (cf. Bird and Liberman, 2000). The framework thus constrains the way the scheme can be structured. When in doubt about the annotation framework of a project, look at the file syntax which most often reflects quite clearly the internal representation.

only a simple string whereas Anvil's track elements hold a number of attribute-value pairs. Partitur was meant to be a file exchange format and has no generic tools for coding.

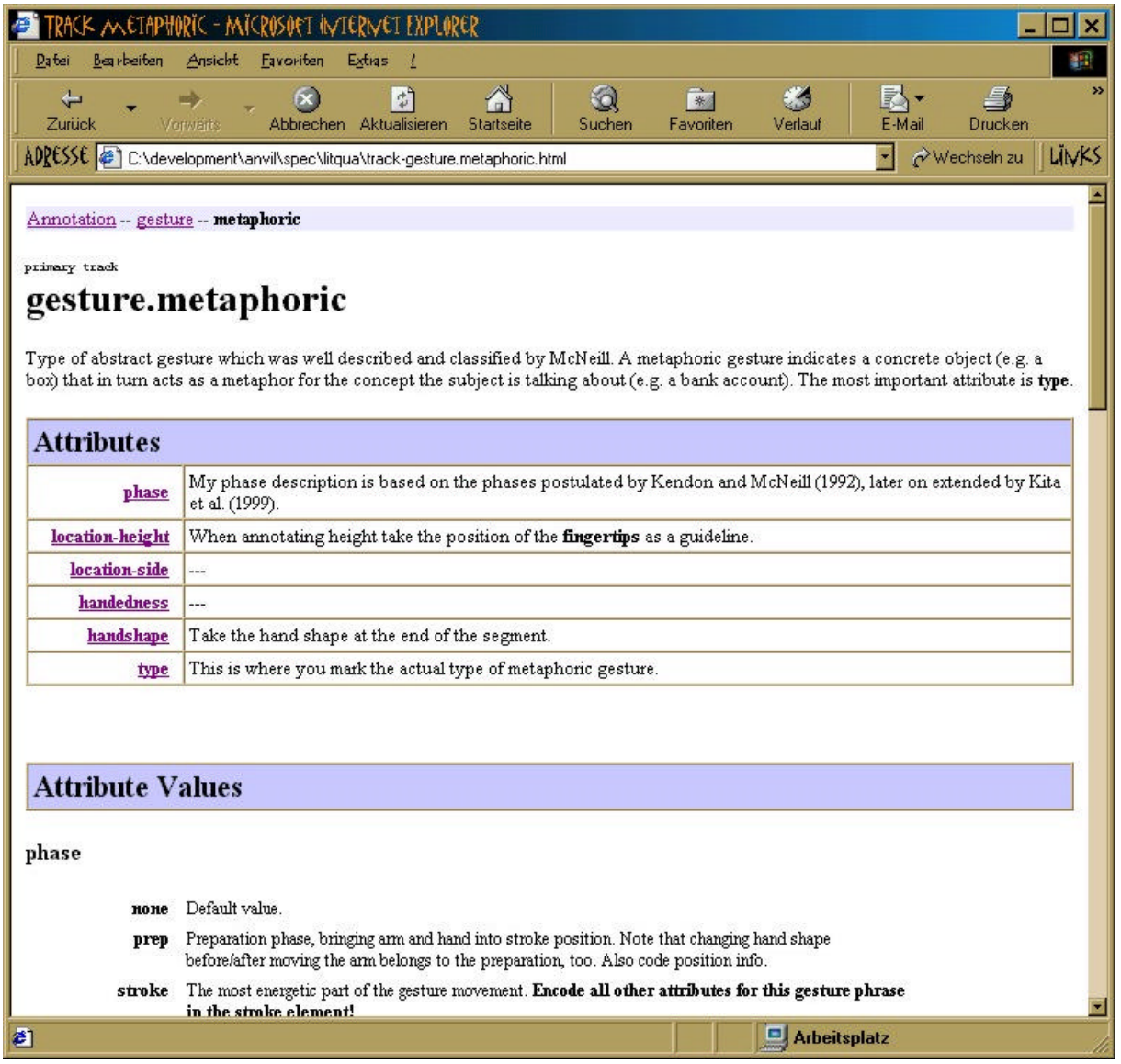


Figure 2.8. Sample page of generated coding manual.

MATE[7], cf. Section 6, is an EU project that aimed at producing a generic workbench for all kinds of linguistic coding (a) and data retrieval (b). Schemes (1) from different projects were taken as models, different file formats (e.g. Partitur) are supported in an XML-based approach. Although quite powerful retrieval concepts were developed the tool as a whole is not available in a stable version where coding (a) can be practically pursued.

The ATLAS tools, cf. Section 3, are based on the concept of annotation graphs (2). They are designed to allow coding (a) and data retrieval (b). Different schemes (1) should be compatible to this framework. The overall design philosophy allows the development of tools interacting with ATLAS on different levels, either on the data level (files) or on the logical level (API).

CAVA[8], cf. Section 7, was developed at the MPI for Psycholinguistics at Njimegen and offers tools for annotation (a), retrieval (b) and analysis (c). CAVA's components run on different platforms (Mac, PC) and use special hardware. It is currently being replicated in an extended, Java-based version called

[7] see http://mate.nis.sdu.dk

[8] see http://www.mpi.nl/world/tg/CAVA/CAVA.html

EUDICO (EU project, cf. Brugman et al., 2000) which is not yet available apart from a limited demo version with only viewing facilities, cf. Section 7.

Akira is a video analysis tool developed at the University of Mannheim for film studies and allows multi-track annotation (1). Its advantages lie in the intuitive user interface. Unfortunately, Akira stores annotation data as binaries which makes further processing difficult. Other systems from related areas, e.g. VideoAS developed at the University of Saarland for media psychological investigations, have a similar problem in that they are too focused on a specific task and do not produce files in a workable format.

2.7 Conclusion

Anvil offers an easy-to-use graphical tool for video annotation. It is being used for investigating nonverbal behaviour and has been developed from a very pragmatic point of view. Its platform-independence, the generic multi-track approach and the use of the universal XML syntax for data storage makes it suitable for a wide range of research applications.

There are many shortcomings some of which will be fixed in the future. Currently, the specification has to be created manually, i.e. with an ASCII text editor (or with an XML editor), where one should provide a specification editor. The user interface offers only the time-aligned view. Anvil could provide other views, for instance a sequential view over all tracks, as well as navigation support like search facilities. The annotation framework is limited, too. It does not allow hierarchical structures to be encoded (e.g. syntax trees). Arbitrary links between elements of different tracks should be possible.

The most important goal, however, is the addition of analysis capabilities, although many institutes will have their own ideas of what analysis to run over the data. Analysis needs data retrieval functions. These do already exist internally but have to be made accessible by some kind of interface (e.g. a Java API).

2.8 References

Bird, Steven and Liberman, Mark (2000). "A Formal Framework for Linguistic Annotation". In: *Speech Communication*.

Brugman, H., Russel, A., Broeder, D. and Wittenburg, P (2000). "EUDICO, Annotation and Exploitation of Multi Media Corpora". LREC 2000 Workshop, Athens.

Cassell, J., Bickmore, T., Campbell, L. and Vilhjálmsson, H. (2000). Human Conversation as a System Framework: Designing Embodied Conversational Agents. In: Cassell, J., Sullivan, J., Prevost, S. and Churchill, E. (eds.) *Embodied Conversational Agents*. Cambridge: MIT Press, pp. 29-63.

McNeill, D. (1992). *Hand and Mind: What Gestures Reveal about Thought*. Chicago: University of Chicago Press.

Kita, S., van Gijn, I. and van der Hulst, H. (1998). "Movement phases in sign and co-speech gestures, and their transcription by human coders". In: Wachsmuth, I. and Fröhlich, M. (eds.) *Gesture and Sign Language in Human Interaction*, Springer, pp. 23-35.

Schiel, F., Burger, S., Geumann, A. and Weilhammer, K. (1998). "The Partitur Format at BAS". In: *Proceedings of the First International Conference on Language Resources and Evaluation*, Granada, Spain.

Wahlster, W., ed. (2000). *Verbmobil: Foundations of Speech-to-Speech Translation*. Springer.

2.9 Appendix

This is an short excerpt of a sample annotation to show the syntax of an anvil annotation file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<annotation>
  <head>
    <specification src="C:\development\anvil\spec\litqua.xml" />
    <video src="c:\Promotion\LitQua\quicktime\lq3-kara.mov" />
  </head>

  <body>
```

```
<track name="trl" type="primary">
  <el index="0" start="0.501372814" end="0.686400651">
    <attribute name="token">ich</attribute>
  </el>
  <el index="1" start="0.7578307980000001" end="1.320613741">
    <attribute name="token">denke</attribute>
    <attribute name="emphasis">strong</attribute>
  </el>
  <el index="2" start="1.8141310210000001" end="2.212828874">
    <attribute name="token">dass</attribute>
  </el>

  ...

</track>

<track name="ling.rst" type="span" ref="trl">
  <el index="0" start="0" end="1">
    <attribute name="relation1">attribution</attribute>
    <attribute name="direction1">forward</attribute>
  </el>
  <el index="1" start="2" end="7">
    <attribute name="relation1">evaluation</attribute>
    <attribute name="direction1">forward</attribute>
  </el>

  ...

</track>

<track name="gesture.beat" type="primary">

  ...

  <el index="5" start="46.159999847" end="46.560001373000006">
    <attribute name="phase">prep</attribute>
    <attribute name="motion-dir">vertical</attribute>
    <attribute name="location-side">outer-right</attribute>
    <attribute name="location-height">chest</attribute>
    <attribute name="handedness">right</attribute>
    <attribute name="handshape">halfopen-flat</attribute>
  </el>

  ...

</track>

<track name="gesture.deictic" type="primary">

  ...

  <el index="1" start="5.840000152" end="6.159999847000001">
    <attribute name="phase">stroke</attribute>
    <attribute name="where">space</attribute>
    <attribute name="location-height">head</attribute>
    <attribute name="location-side">right</attribute>
    <attribute name="handedness">right</attribute>
```

```
        <attribute name="handshape">index-out</attribute>
       <comment>
         refering to "Buch"
       </comment>
      </el>

      ...

    </track>

    ...

  </body>
</annotation>
```

3 ATLAS

3.1 Introduction

ATLAS (Architecture and Tools for Linguistic Analysis Systems) is a US project formed by NIST, the LDC, and MITRE in order to investigate the creation of tools and formats to address the needs of flexible and extensible annotation formats and tools that can accommodate and facilitate evolution and synergy. The project is an ongoing effort at the organisations working on the project and is funded by various other projects, which will make use of the formalism and tools. The project started in the beginning of 2000. It is expected that the major effort will be complete during the year 2002 and that it will become an open-source evolving resource that NIST will administer.

ATLAS aims at developing a general architecture for annotation including a logical data format, an API and tool-set, and a persistent data representation. The architecture needs to be modular, flexible and extensible and facilitate the exchange and reuse of existing language resources. Such resources are to be moved in and out of the ATLAS framework via conversion routines, and therefore ATLAS also aims at acting as an interlingua for language corpora. Furthermore, the architecture is expected to provide a platform for the extension of such corpora and the development of new ATLAS native resources. The final goal of ATLAS is that it will serve as a conduit to enable the greater flow of language resources throughout the language research community.

ATLAS is not a tool in itself, but should rather be viewed as an underlying annotation data management component used by a number of annotation tools. ATLAS is to cover the domain of any form of linguistics, which involves annotation of signals, and should eventually ease the process of developing linguistic applications.

The ATLAS developers' reasons for creating ATLAS correspond to those behind the development of the MATE Workbench, i.e. they believe that we live in a world with an increasing variety of technology capabilities and a corresponding wealth of research corpora, yet these capabilities and corpora are domain-specific, task-specific and often even site-specific. There is an increasing need for annotated corpora, which are designed for diverse use. Linguistic annotation is expensive to produce and adapting existing linguistic corpora for a new use may require a great deal of time and extensive re-engineering. Designing annotation corpora for reuse and enabling existing corpora to be used for new purposes will reduce costs as well as provide "instant" resources for research. There is therefore a great need for flexible and extensible annotation formats and tools that can accommodate and facilitate evolution and synergy.

The partners in the ATLAS project overlap with those in the TalkBank project (cf. Section 13), and there also seem to be close links between the two projects. For instance the TalkBank project funds LDC's work in the ATLAS project, whereas NIST is funded from NIST-internal money and from the DARPA TIDES program. The DARPA TIDES program also funds MITRE.

No demo or final product is available yet. However, a prototype of a simplified version working with only linear signals should be available from the LDC website (http://www.ldc.upenn.edu/), but could not be found at the time of writing (December 2000). NIST/MITRE is developing a more comprehensive version, which should be available from the NIST website (http://www.nist.gov/) in the early spring of 2001. The demos of ATLAS will be freely available. The present review is based on the ATLAS web site, comments from the developers of ATLAS, and (Bird et al. 2000).

3.2 Software design

ATLAS is built as an object-oriented framework providing a hierarchy of classes as well as extension points to which users can plug in their specific versions of ATLAS components so as to integrate them in the general architecture (Figure 3.1), while providing specialised behaviour for their application. The framework has a modular organization and provides different services to developers of linguistic applications.

The current services are: abstract model implementation allowing manipulation of annotation data in the architecture, transparent access to signals regardless of their type is being worked on, persistence

framework providing an abstraction to the physical storage of annotations allowing different persistence modalities to be used transparently (currently the modality of choice is XML but a database access module is being worked on and stream based persistence for communication of ATLAS applications across a network is being investigated).

The framework is based on an abstract model that is derived from the annotation graph model (Bird et al. 2000) but extends it to overcome its limitations. The ATLAS Interchange Format (AIF) is a concrete instantiation of this abstract model, i.e. it is a means to concretely exchange and store annotation data that fit the abstract model. It is an XML-based representation. AIF is meant to provide a concrete way to express ATLAS' annotation model thus allowing exchange of data between ATLAS-compliant applications. Annotations in themselves are manipulated via ATLAS' APIs and can optionally be read from and written to AIF-based files. A query framework will be provided later. A set of reusable widgets to be used in applications will also be provided later.

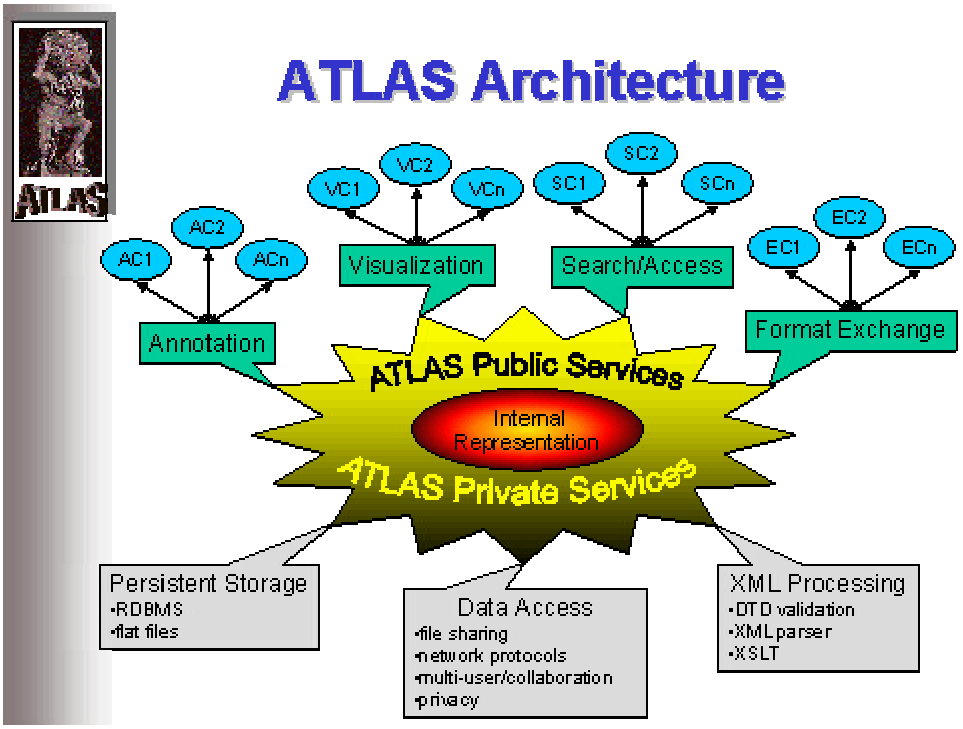


Figure 3.1. ATLAS architecture.

ATLAS is to consist of three levels: application, logical and physical (cf. Figure 3.2).

The implementation languages are Java, C++ and Tcl. ATLAS will in the future provide scripts to migrate current annotation schemes to ATLAS.

3.3 Platform requirements

ATLAS is planned to run on any platform that supports Java, C++ and Tcl implementation. No special hardware besides a sound card to enable access to signal data should be needed.

3.4 Interface

The software will not have a specific interface since it is not an application but a framework.

3.5 Functionality

The idea of ATLAS is that functionality will be up to the user who uses the framework to create his own functionalities, which makes the list of functionalities virtually infinite. The ATLAS developers have not offered any information on which functionalities the framework is born with.

The developers expect that the advantages of ATLAS will be that it is generic and not tied to a particular annotation scheme and/or signal modality. It will be extensible, so that people can create new annotation schemes and new classes of signals and have them handled by ATLAS compliant applications without problems. Furthermore, it will provide the basic services needed to create full-fledged linguistic applications.

The disadvantage of ATLAS is that it is not a direct application and is therefore not highly usable in itself, but only in combination with other tools.

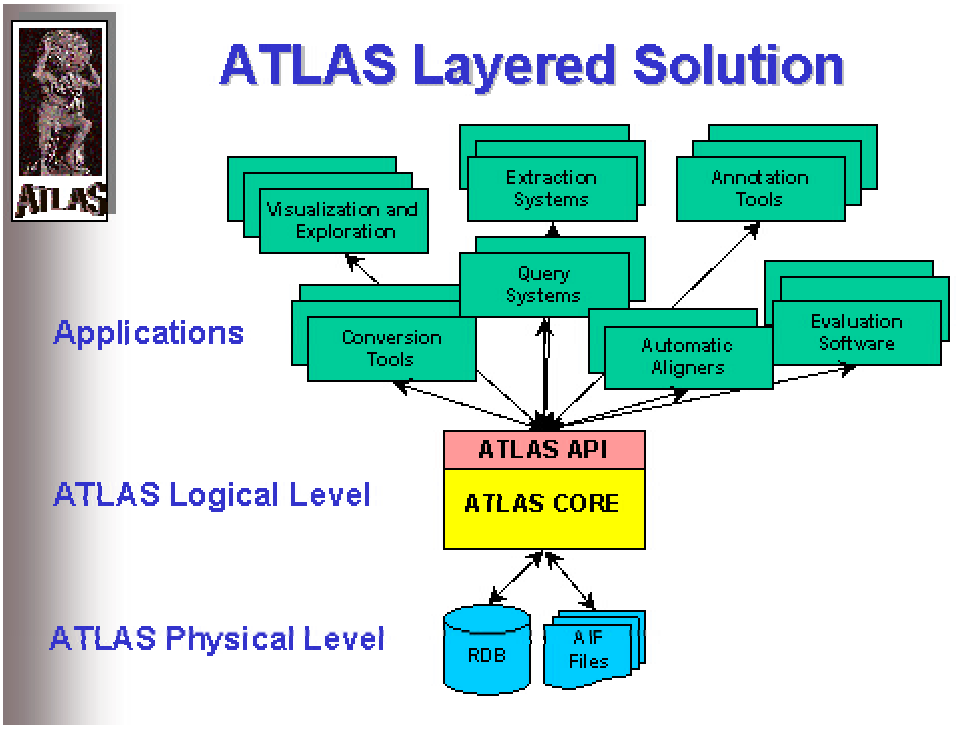


Figure 3.2. ATLAS layers. AIF = ATLAS Interchange Format. RDB = Resource database. API = Application Programming Interface.

3.6 Conclusion

Since no demos or actual product has been available from the ATLAS project during the review period, only the ideas behind ATLAS can be evaluated. The final goal of ATLAS, which is that ATLAS will serve as a conduit to enable the greater flow of language resources throughout the language research community is a good idea but ambitious. As long as it is not possible to get hands-on experience with project results, we can only wait and see to which extent the outcome will meet the ambitions.

3.7 References

ATLAS: http://www.itl.nist.gov/iaui/894.01/atlas/ and

http://www.itl.nist.gov/iaui/894.01/atlas/resources.html and

http://www.itl.nist.gov/iaui/894.01/atlas/new/develop/aif.html

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

LDC website: http://www.ldc.upenn.edu/

NIST website: http://www.nist.gov/

Publications on ATLAS: http://www.ldc.upenn.edu/sb/home/publications.html#lrec00-atlas

4 CLAN

4.1 Introduction

The acronym CLAN stands for Computerized Language Analysis. It is a program that is designed specifically to analyze data transcribed in CHAT, the format of the Child Language Data Exchange System (CHILDES).

CLAN was developed for research purposes and is distributed free of charge by the Department of Psychology at Carnegie Mellon University. It was written by Leonid Spektor at Carnegie Mellon University. CLAN allows the user to perform a large number of automatic analyses on transcript data. The analyses include frequency counts, word searches, co-occurrence analyses, mean length of utterance (MLU) counts, interactional analyses, text changes, and morpho-syntactic analysis.

CLAN can be retrieved from http://childes.psy.cmu.edu using a Web Browser. The present overview is based on indirect experience of CLAN tools and on comments from a person working in the CHILDES project for many years.

4.2 Software design

CLAN has a modular structure, where different program tasks have been distributed over multiple modules, all located in the bin directory of CLAN. The Commands window is the centre of all modules, and controls creation, management and linking of coded transcripts through the Editor, as well as activation of analysis tasks to be performed on coded transcripts and redirection of their output through the Analysis commands.

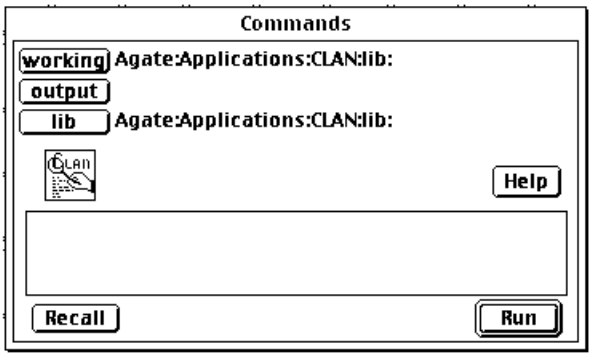


Figure 4.1. CLAN commands window.

All working files which are specific of a certain project/task should preferably be located in the same working directory. Missing working files are looked for in the lib directory. The setting of both working and lib directories is done through the CLAN commands window, which comes up by double-clicking on the CLAN icon. Figure 4.1 shows what the **Commands** window looks like. The window remains active in the background until the program is terminated. The current version (autumn 2000, release number not available) uses a graphic user interface and runs on both Macintosh and Windows machines. Earlier versions also ran on DOS and Unix without a graphic user interface.

4.3 CHAT files

Once the lib and working directories have been set up through the commands window, a CLAN user needs first to create CHAT files. CHAT files are transcript files, containing CHAT codes which serve the purpose of conveying single or multiple levels of linguistic annotation. Specifications concerning CHAT annotation schemes can be found in the CHAT manual, available at http://childes.psy.cmu.edu.

The three main components of a CHAT file are the *file headers*, the *main tier*, and the *dependent tiers*. A header line is a line of text that gives information about the participants and the setting of a given dialogue transcript. All headers begin with @. Here is a simple example of a file in Spanish:

@Begin
@Participants: CHI Target_Child, MOT Mother
*CHI: hasta mañana
*MOT: ¿qué? creo que sí.
@End

Linguistic annotation in CHAT is articulated over one or more tiers. The first tier, called the main tier, contains the basic transcript, and additional tags which appear interspersed with the actual transcription text. Each main tier is preceded by '*', as illustrated by the example above.

More articulated and comprehensive annotations are marked up at secondary tiers, called dependent tiers, which are shown in the lines immediately following the main tier. In the following example, the main line is accompanied by two dependent tiers, one containing information about the speech acts conveyed by the utterance, and the other specified for morpho-lexical features, that are aligned with word tokens in the main line.

*MOT: well go get it!
%spa: $IMP $REF $INS
%mor: ADV|well V|go&PRES V|get&PRES PRO|it!

4.4 The Editor: a bird's eye view

CLAN includes an editor that is specifically designed to work cooperatively with CHAT files.

Figure 4.2 shows what a new text window looks like on the Macintosh:

Figure 4.2. CLAN editor window.

One can type into this editor window just as in any other text editor. The editor works in several modes, each corresponding to a different function. The selection of different modes is done from the **Mode** pull-down menu. There are several editor modes.

1. **Text Mode** In Text Mode, the editor functions as a basic ASCII editor. To indicate that the editor is in Text Mode, the bar at the bottom of the editor window displays [E][Text]. To enter Text Mode, one has to uncheck the CA Mode and CHAT Mode buttons on the Mode pull-down menu.

2. **CHAT Mode** In CHAT Mode, the editor facilitates the typing of new CHAT files and the editing of existing CHAT files. If the file has the extension .cha, the editor will automatically place the file into CHAT Mode. To indicate that the editor is in CHAT Mode, the bar at the bottom of the editor window displays [E][CHAT].

3. **CA Mode** As an alternative to CHAT, you may wish to use CA format. This mode is intended for use by researchers working in the field of Conversation Analysis and should not be used for data in the CHILDES database or for data that will be added to the CHILDES database. If the file has the extension .ca, the editor will automatically place the file into CA Mode. To indicate that you are in CA Mode, the bar at the bottom of the editor window displays [E][CA].

4. **Coder Mode** In Coder Mode [C], the editor provides a systematic interface for inserting codes into a new coding line from a predefined coding menu.

5. **Disambiguator Mode** When the editor is in Disambiguator Mode, it is possible to disambiguate the output of the MOR program on the %mor line.

6. **Sonic Mode** In Sonic Mode (with the waveform displayed), you can link the transcript in your file to a digitized sound file. A wave form is displayed at the bottom of the screen and the beginnings and ends of sounds are indicated in the transcript with millisecond values. Once these links are made, sounds may be directly played from the transcript.

7. **Continuous Playback Mode** There are two continuous playback modes — one for sonic playback and the other for movie playback. They operate in similar ways. In Continuous Sonic Playback Mode, the waveform display is turned off and the machine plays back the entire transcript, one utterance after another, while moving the cursor and adjusting the screen to continually display the current utterances. In Continuous Movie Playback Mode, the video is played as the cursor highlights utterances in the text.

8. **Video Mode** Just as it is possible to link transcripts to digitized audio, it is also possible to link them to digitized video with audio. One can use the options in the Mode pull-down menu to turn these modes on and off.

In what follows we will briefly overview the basic functionalities of the CLAN editor in the CHAT and CODE modes, and then cover the Video and Sonic modes in more detail.

4.4.1 Basic Functionalities in the CHAT and CODE modes

The CHAT mode provides the user with functionalities specifically designed to work cooperatively with CHAT files.

The editor in CHAT mode can hide one or more dependent tiers. It can also run a consistency check over the current CHAT file through the CHECK command.

On entering the CODE mode, the annotator is asked to load a *codes file*. A codes file contains a list of all codes which are specific of one or more dependent tiers. Codes lists can be arranged hierarchically in the codes file. The CLAN editor in the CODE mode allows efficient menu-driven browsing of the codes hierarchy specified in the loaded codes file and quick selection of the contextually appropriate set of codes from the browsed hierarchy.

The CLAN editor in the Code mode is able to look at the relevant information encoded at each header line and to structure this information into on-line help menus. For example, when a CHAT file is opened, the editor looks at each of the participants declared for the file and inserts their codes into the Tiers menu.

In addition to the various modes for efficient text editing and coding, the CLAN editor provides methods for linking the transcript to digitized audio and video. These modes are called Sonic Mode and Video Mode. We will begin with a description of Sonic Mode.

4.4.2 Sonic Mode

Use of the Sonic Mode requires availability of a digitized audio file. Once the file has been selected from the CLAN editor in the Sonic Mode, the waveform comes up, starting at the beginning of the file. Several functions are available at this point (see Figure 4.3 below):

1. **Sound playing from the waveform** the cursor can be dragged over a segment of the waveform to highlight it. Upon releasing the mouse, the segment will play. As long as it stays highlighted, one can replay it by holding down the shift key and clicking the mouse. At this point, it does not matter where the cursor is positioned.

2. **Waveform demarcation** the borders of a highlighted region can be moved by holding down the shift key and clicking the mouse to place the cursor at the place to which region is intended to be moved. This method can be used to either expand or contract the highlighted region.

3. **Transcription** While working with the wave form, one can repeatedly play the sound by using shift-click. This will help the user to recognize the utterance being transcribed. One can then go back to the editor window and type out the utterance that corresponds to the highlighted segment.

4. **Linking** When the highlighted waveform is believed to correspond correctly to the utterance being transcribed, a bullet can be inserted upon clicking on the "s" button to the left of the waveform. This bullet contains information regarding the exact onset and offset of the highlighted segment (see Figure 4.3).

5. **Changing the waveform window** The +V and -V buttons on the left allow the user to increase or decrease the amount of time displayed in the window. The +H and -H buttons allow the user to control the amplitude of the wave form.

6. **Scrolling** At the bottom of the sound window is a scroll-bar that allows the user to move forward or backward in the sound file.

7. **Waveform activation**. In order to highlight the section of the waveform associated with a particular utterance, one needs to triple-click on the bullet following the utterance one wants to replay. The waveform will redisplay. Then one can re-play it by using shift-click.

8. **Expanding and hiding the bullets**. Exact temporal references that are hiding inside the bullet symbols can be displayed by typing Es -A to expand them. Typing Es -A again will hide them again (see Figure 4.3).

9. **Time duration information**. Just above the waveform, the editor mode line is displayed. Clicking on this line displays additional numbers, namely i) the beginning and end time of the current window in seconds, ii) the position of the cursor in hours:minutes:seconds.milliseconds, and iii) the beginning and end of the current selection in seconds. Sampling rate information for the audio file can be displayed by clicking on the editor mode line once more.

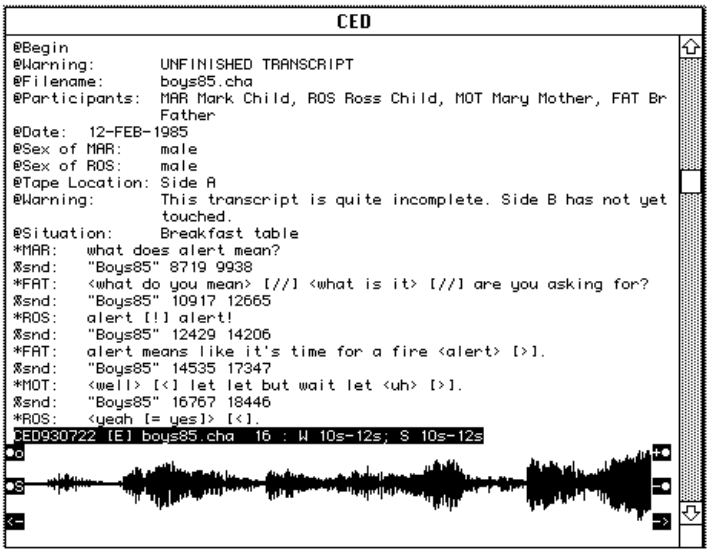


Figure 4.3. CLAN Sonic mode.

4.4.3 Video Mode

Video Mode works very much like Sonic Mode. It presupposes availability of a digitized video file. To play a file that has already been linked to a movie, the annotator must open the transcript file and click on the bullet following the utterance you wish to play. This will open the movie and play the segment. One can also use Continuous Playback Mode to play all the segments in the file. Moreover it is possible to link a transcript to a movie or create a new transcript that is linked to a movie.

4.4.4 Other functionalities (analysis commands)

CLAN is equipped with a large number of commands/functions for the automatic linguistic annotation of texts, and for the quantitative analysis and browsing of annotated texts. A complete list of them would largely exceed the space available here. Suffice it to say that the battery of functions ranges from advanced boolean search facilities (*combo*), to tools for morphological annotation, interactional analysis, co-occurrence analysis, developmental analysis (in terms of Development Sentence Score) and frequency counts of various kinds. For further information, interested readers are referred to the

CLAN manual, a clear introductory guide to CLAN professional use (available at http://childes.psy.cmu.edu).

4.5 Conclusion

CLAN has continuously been developed and refined since 1981 to address the specific requirements of a large research community, actively working in the fields of child language and language acquisition, adult conversational interactions, sociological content analyses, and language recovery in aphasia. It represents a very good example of a practical annotation tool developed on the basis of de facto scientific standards and co-operative feedback, both coming from consolidated practices of analysis of text corpora and digitized video and audio material.

Some of CLAN design choices reflect these operational and scientific concerns. For example, storing all annotation tiers in the same matrix file seems to respond well to the working practice of one single annotator working on a comparatively small corpus of richly annotated dialogues, which need to be searched efficiently at more than one level of annotation. Although the CLAN editor is capable of providing user-oriented views of the same material, however, it is difficult to see how different annotators can simultaneously mark up the same matrix text at different annotation tiers. Furthermore, CLAN supported annotation schemes are rather skeletal, and the issue of their scalability to more complex and articulated annotation schemes is far from trivial.

In spite of these shortcomings, CLAN's simple and user-friendly interface, its lean and effective implementation and its wealth of commands for automatic text analysis make it a still unrivalled example of a powerful annotation tool dealing with multimodal data. Finally, the success history of its development and dissemination (in about 60 different countries) makes it a remarkable model of highly co-operative design and specification effort.

4.6 References

http://childes.psy.cmu.edu

http://atila-www.uia.ac.be/childes/ (European CHILDES mirror site)

5 CSLU Toolkit

5.1 Introduction

The CSLU Toolkit was developed at the Center for Spoken Language Understanding, a research centre at the Oregon Graduate Institute of Science and Technology (OGI) in Portland, Oregon (USA). (The Toolkit is being jointly maintained and developed by CSLU and the Center for Spoken Language Research (CSLR) of the University of Colorado, which branched off from CSLU in 1998.[1])

The Toolkit is designed for both research and educational purposes in the area of speech and human-computer interaction.

The core components of the Toolkit cover the domains of speech recognition, text-to-speech synthesis, waveform analysis, facial animation technologies, dialogue modelling, and natural language understanding. A general software development environment is also provided. In addition to these areas, the Toolkit incorporates tools developed at the Perceptual Science Laboratory of the University of California at Santa Cruz, for conducting and statistically analyzing perceptual experiments.[2]

In addition to supporting research in spoken language technologies, the Toolkit is intended for educational purposes such as language training (see Cole et al. 1999, Stone 1999, Connors et al. 1999).

The currently available version of the Toolkit is the second beta release, 2.0b2. A final version 2.0 release was originally scheduled for August 2000, but has been delayed.[3]

The entire Toolkit is freely downloadable via HTTP and FTP.[4] Registration and compliance with the license are required, which limits use of the Toolkit to non-commercial, specifically "educational, research, personal or evaluation purposes."[5] The complete installation includes both compiled executable files and source code for all of the Toolkit's components. In addition, the installation comes with demonstrations and tutorials of one of the principal components of the Toolkit, the *Rapid Application Developer* (RAD); these are also available online.[6] For another of the Toolkit's main components, *Speechview*, extensive documentation and a tutorial are available on the Toolkit's homepage.[7] CSLU also hosts a newsgroup devoted to questions about the Toolkit.[8]

In addition to the Toolkit's native components, the complete installation also contains version 1.3.1 of the Festival text-to-speech system, originally developed at the University of Edinburgh,[9] augmented with several extensions developed at OGI. Binary files and source code for Festival and the OGI extensions are also included. In order to support application development by users, the Toolkit also contains version 8.0 of the Tcl/Tk programming language, including source code and binaries.

For this review we downloaded and installed version 2.0b2 of the Toolkit, but initially we faced hardware limitations and have also had some problems with the software, which has prevented a thorough examination of some of the components (in particular, of the OGIsable GUI discussed below). Much information, however, is available online, both at the various toolkit websites (CSLU, CSLR, PSL) and in a number of papers (see references). [10]

---

[1] See http://cslr.colorado.edu/.

[2] See http://mambo.ucsc.edu:80/psl/tools/.

[3] See http://cslu.cse.ogi.edu/toolkit/news.html.

[4] See http://cslu.cse.ogi.edu/toolkit/download/index.html.

[5] See http://cslu.cse.ogi.edu/toolkit/download/license.html.

[6] See http://cslu.cse.ogi.edu/toolkit/docs/2.0/apps/rad/movies/introduction.avi,
http://cslu.cse.ogi.edu/toolkit/docs/2.0/apps/rad/start/index.html, and
http://cslu.cse.ogi.edu/toolkit/docs/2.0/apps/rad/tutorials/index.html and
http://cslr.colorado.edu/toolkit/flash/Tutorial0.swf.

[7] See http://cslu.cse.ogi.edu/toolkit/docs/2.0/apps/speechview/index.html.

[8] See news://cslu.cse.ogi.edu/cslu.toolkit.

[9] See http://www.cstr.ed.ac.uk/projects/festival/.

[10] I would like to thank Ulrich Heid for numerous discussions of and suggestions concerning the content and form of this report.

5.2 Software design

The Toolkit "provides a modular, open architecture supporting distributed, cross-platform, client/server-based networking. It includes interfaces for standard telephony and audio devices, and software interfaces for speech recognition, text-to-speech and animation components. This flexible environment makes it possible to easily integrate new components and to develop scalable, portable speech-related applications (Sutton et al. 1998)."

The core functionalities of the Toolkit's main components, the Rapid Application Developer and the speech recognition tools, as well as the programming environment, have an underlying implementation as a set of C libraries.[11] These core functionalities were then re-implemented in an extension of Tcl to facilitate writing, debugging, and modification.[12] On top of the core implementation graphical user interfaces written in Tcl/Tk were added.[13] The Festival TTS system included in the Toolkit, and its OGI extensions, are written in C++ and Scheme.

The OGIsable GUI, which is an auxiliary component of the Toolkit (see Section 5.4.1 and footnote 21 below), requires installation of the Windows distribution of TclExpat (a Tcl interface for the XML parser expat), which is available as shareware from http://www.zveno.com/zm.cgi/in-tclxml.

5.3 Platform requirements

"The CSLU Toolkit currently runs under Windows 95, 98, NT and 2000 beta on Intel (or compatible) processors only."[14] Minimum Requirements:

- Intel Pentium machine
- 200 MHz processor
- 64 MB Ram (128 preferred)
- Windows 98 or NT 4.0*
- Microphone / Speakers (sound-blaster compat.)

"A minimal installation will download 25 Mb of data and the "typical" (recommended) installation 28 Mb. A complete download of the toolkit and all of the optional modules and TTS voices is around 88 Mb."[15]

5.4 Interface

5.4.1 Description of interface design and usability of the tool

The principal user application of the Toolkit is the Rapid Application Developer (RAD). "This is a tool for creating structured dialogues between users and the computer. With it you can create a wide variety of interactive programs that run both over the telephone and on the desktop. In RAD you drag and drop dialogue states onto a canvas, connect them together, and configure them to do things like play audio files, create animated text-to-speech, recognize spoken language, or display images."[16] The RAD user interface consists of four windows (Figure 5.1). These windows contain the RAD canvas; a toolbox of RAD objects, which can be clicked on and dragged onto the canvas; an animated agent, Baldi (developed at PSL, see e.g. (Cohen et al. 1998)), which produces synthesized speech; and a caption box, containing the text of Baldi's speech.

---

[11] See http://cslu.cse.ogi.edu/toolkit/old/old/version2.0a/documentation/csluc/Cdoc.html.

[12] See http://cslu.cse.ogi.edu/toolkit/old/userguide/cslush/cslush.html, http://cslu.cse.ogi.edu/toolkit/old/old/version2.0a/documentation/devsh/develop.html.

[13] See http://cslu.cse.ogi.edu/toolkit/old/old/documentation/cslurp/wincslurp/ newmanual.html.

[14] This quote and the one below are from http://cslu.cse.ogi.edu/toolkit/download/index.html; the list of minimum requirements is from http://mambo.ucsc.edu/psl/milass2.ppt.

[15] This information is mistaken or outdated; the complete installation that was downloaded for this review comes to more than 280 MB (and this does not include the OGIsable GUI, see below).

[16] http://cslu.cse.ogi.edu/toolkit/docs/users.html.

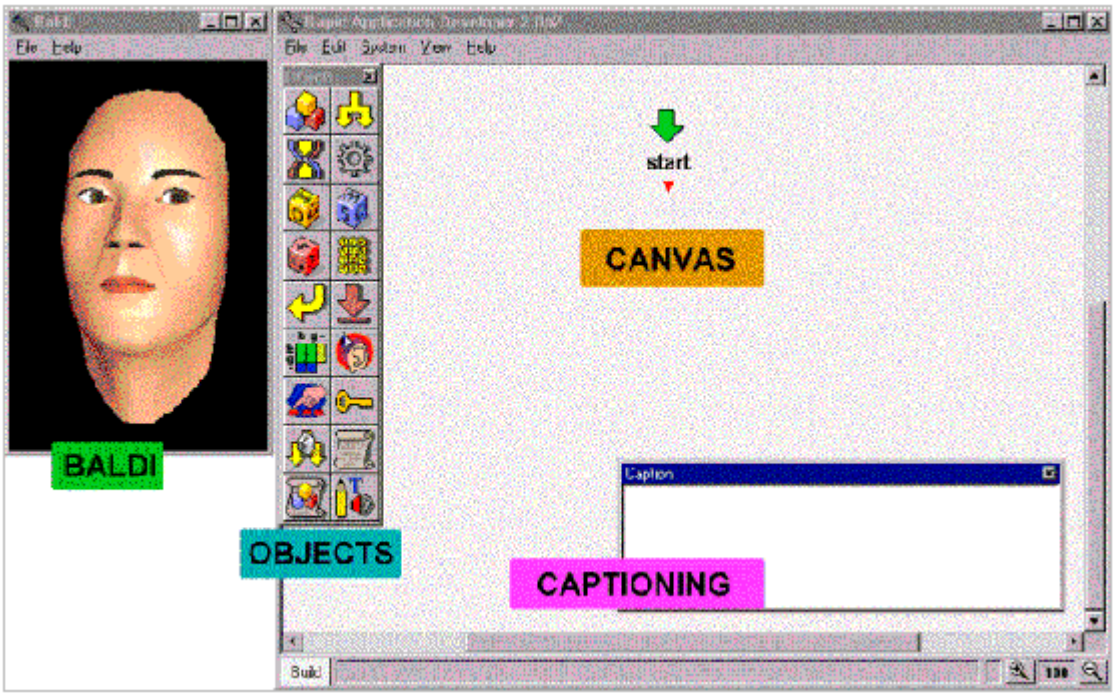


Figure 5.1. RAD User Interface.

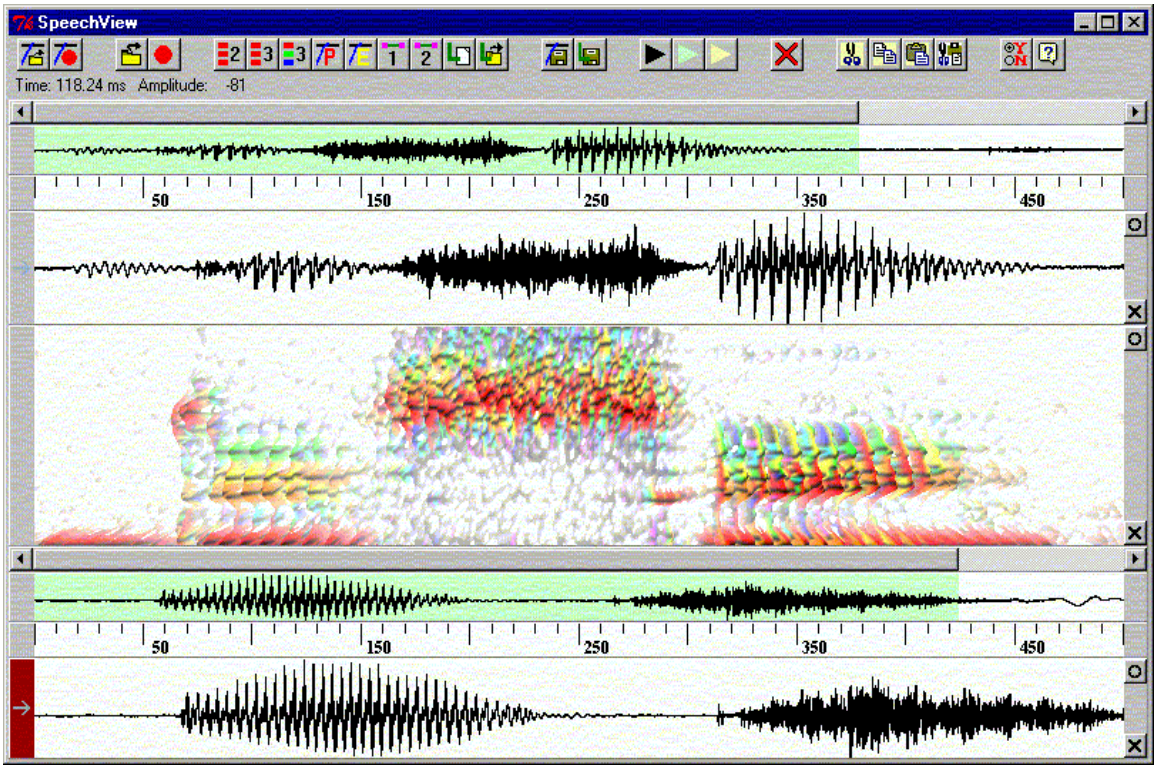


Figure 5.2. SpeechView Window.

The two other main applications of the standard Toolkit installation are SpeechView (Figure 5.2) and BaldiSync (Figure 5.3). The SpeechView tool is "used for displaying, playing, and editing speech waveforms. It can also display spectrograms and other types of data related to speech waveforms such as pitch and energy contours, neural net outputs, and phonetic labels. SpeechView can read previously created waveform files; it can also record new waveforms and save them to disk. The basic

functionality provided by SpeechView is used in other CSLU Toolkit components such as BaldiSync and the Rapid Application Developer."[17]

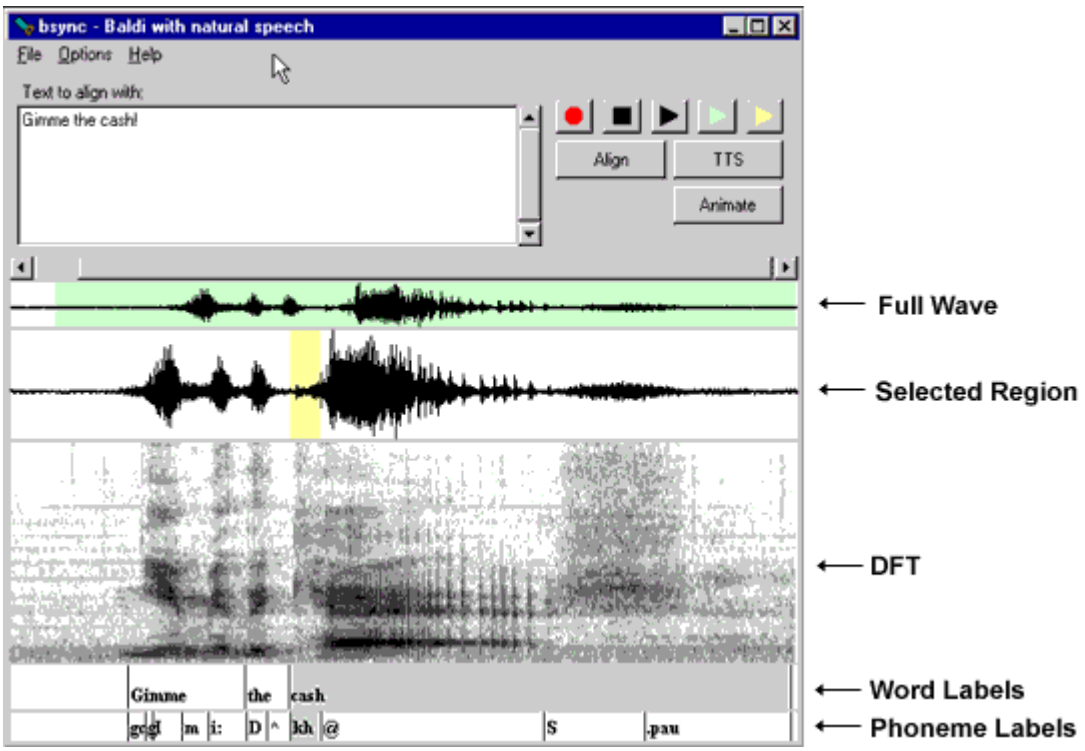


Figure 5.3. BaldiSync Window.

"Baldi Sync is a tool which you can use to view and create facial animation that is aligned with recorded audio. You can use either your own recorded voice or text-to-speech as the audio source. Baldi Sync can read and write wave files, as well as its own special files called sound objects (sobs)."[18]

The Toolkit also comes with a Tcl programming shell called CSLUsh. "CSLUsh provides powerful tools to manipulate wave files, extract features, train and utilize artificial neural networks and Hidden Markov models, recognize spoken utterances subject to grammar and word model constraints, and perform other related activities."[19]

As an additional module to the basic Toolkit, there is a graphical user interface for an extended version of the XML-based Sable TTS markup language, which was developed at OGI and is called OGIsable.[20] "The tool allows the user to highlight text and attach properties to it… The graphical interface improves the expressivity of the markup commands because authors can quickly experiment with combinations of tags to reach a desired result" (Wouters et al. 1999). The screen shots in Figure 5.4, Figure 5.5 and Figure 5.6 show a customizer window with an inventory of the graphic markup symbols, an example of graphically marked-up text, and the corresponding XML source.[21]

---

[17] http://cslu.cse.ogi.edu/toolkit/docs/2.0/apps/speechview/index.html.

[18] http://cslu.cse.ogi.edu/toolkit/docs/2.0/apps/baldisync/index.html.

[19] http://cslu.cse.ogi.edu/toolkit/old/userguide/cslush/cslush.html.

[20] Sable is the markup language for the Festival TTS system. See (Sproat et al. 1997) and http://www.cstr.ed.ac.uk/projects/ssml.html.

[21] These screen shots are of an earlier version of the GUI. They are taken from slides found online at http://mambo.ucsc.edu/psl/milass2.ppt, dated July 1999. The OGIsable markup tool itself is not mentioned on the Toolkit homepage and is not included in the "complete" installation we downloaded, though according to http://cslu.cse.ogi.edu/tts/download/index.html: "A graphical interface to OGIsable is provided in the CSLU Toolkit." However, I found a downloadable version of the tool at ftp://cslu.cse.ogi.edu/pub/wouters/OGIsableGUI-1.3.tar.gz, dated November 1999. In this version of the GUI,

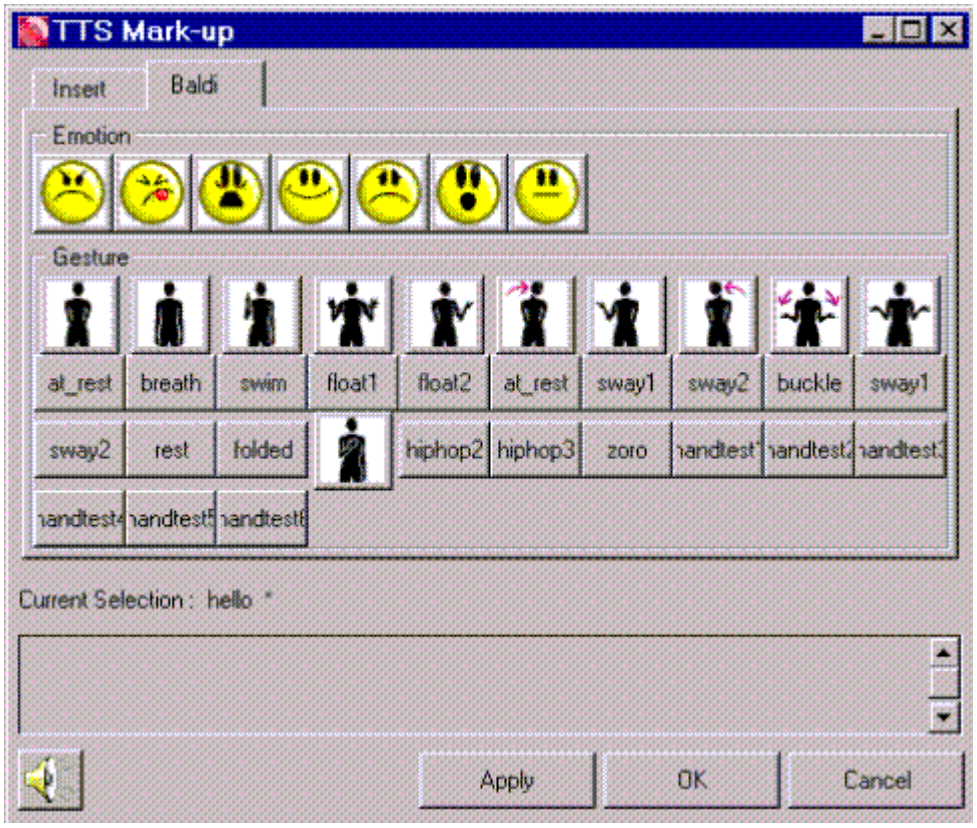


Figure 5.4. OGIsable graphic markup symbols.

5.5 Functionality

The core components of the CSLU Toolkit encompass of broad range of functionalities in the domains covered by the Toolkit (listed above). However, most of these are not directly applicable to multimodal corpus annotation. The two components that do have functionality relevant to the concerns of ISLE are SpeechView and the OGIsable GUI (some of SpeechView's functionality is incorporated into RAD and BaldiSync).

5.5.1 SpeechView

SpeechView displays the waveform of a recorded speech signal and provides various functions for viewing and manipulating this waveform. The waveform can be created from a preexisting recorded sample or generated by making a recording via a microphone connected to the computer. A portion of the complete waveform can be selected (the green shaded part of the small waveforms in Figure 5.2) and given a magnified display (the larger waveform).

These basic waveform displays can be augmented with a number of other displays showing various attributes of the waveform. There may be up to three different spectrograms of the waveform: two- and three-dimensional gray-level displays (the former is seen in Figure 5.3) plus a three-dimensional colour display (Figure 5.2); pitch and energy contour displays; displays of additional one- and two-dimensional objects time-aligned with the waveform, which are user-defined (an example of a two-dimensional object other than a spectrogram is the output of a neural network); and any number of labellings of the waveform, which may either be directly entered by the user or loaded from a

---

the marked-up text and the XML source occupy the upper and lower frames, respectively, of a single window. Also, the customizer window is configured somewhat differently than that illustrated in Figure 5.4. Although I have installed this tool in the Toolkit and tested its functionality (though this was limited because certain problems with the source code were experienced), but I have not yet been able to make screen dumps of its GUIs, hence have used the older images available online for this review.

preexisting file. These nine types of displays are activated by the buttons in the toolbar shown at the top of Figure 5.2. The sum of these displays, including the waveform, is referred to as the wave group. It is also possible to simultaneously display more than one wave group (there are two in Figure 5.2).

In addition to displaying waveforms and their attributes, SpeechView has various functions for playing and editing waveforms. A selected waveform segment can be played either by clicking on the corresponding play button in the toolbar or by clicking on the selected segment itself. It is also possible to remove portions of silence or other segments of the waveform and perform cutting and pasting on portions of the waveform. These functions are also activated by the toolbar.

For purposes of annotation, the label windows provide basic functionality. These windows can be partitioned into cells aligned with relevant segments of the other displays (such as a spectrogram). The cells can be labelled by typing in e.g. words or phonetic transcriptions (at the user's discretion); this is illustrated in Figure 5.3. Note that each label window is independently segmented, which allows representing e.g. hierarchical and overlapping structure. Clicking on a cell selects it (shown in the zoom display by orange shading) and plays that segment of the recording. These selected segments can also be edited.

Many of the properties of the various displays and functions of SpeechView can be controlled by pop-up menus from the graphical user interface or, for more fine-grained tuning, by setting flags in the command line interface (in the CSLUsh shell).

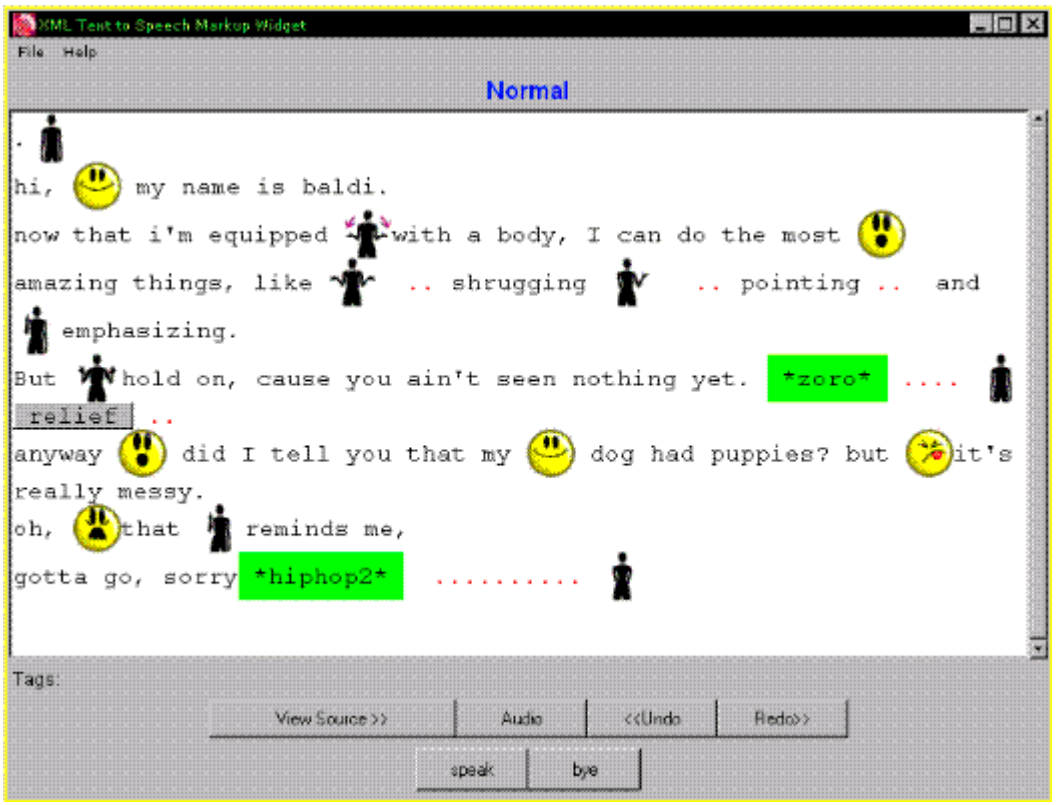


Figure 5.5. OGIsable TTS markup.

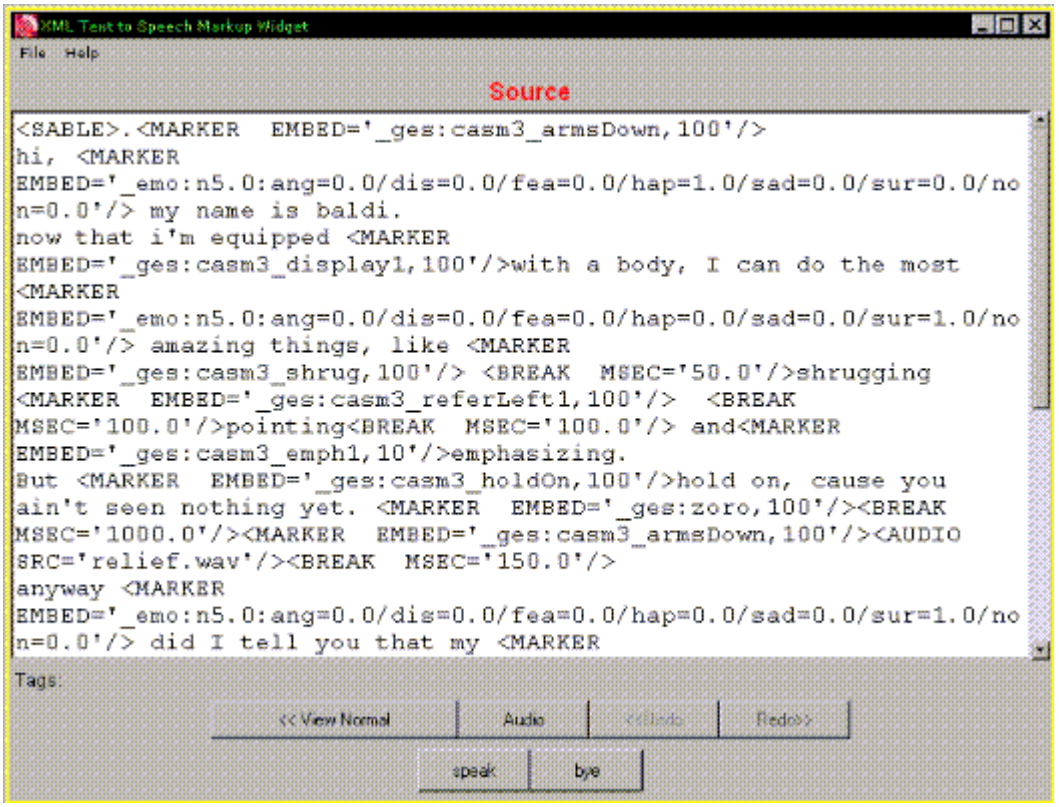


Figure 5.6. OGIsable XML source.

5.5.2 OGIsable GUI

The only specific annotation tool in the Toolkit (though, as noted above, it is not included in the currently available version) is the OGIsable markup GUI. This tool provides various functions for marking up and modifying the spoken output of the input text: Graphical markup is added to the text via the customizer window, either by inserting selected symbols into the text or by directly altering the text appearance (e.g., italics, underlining, color). An update button displays the Sable XML code of the graphical markup. In addition to using the customizer window, both the input text and the Sable markup can be directly edited in their respective windows. The results of the markup can be sent to the Festival TTS engine and spoken by Baldi with appropriate intonation and facial expression.

For markup inserted between text, there are menus for selecting

- the type of intonation break (automatic, none, phrase, sentence, pause with specifiable duration);
- for choosing the voice type with which Baldi speaks (from the voices available);
- for selecting predefined sounds to add to the text (e.g. annoyed, burp, hmm, laugh, sneeze, yawn, etc.);
- and for selecting emotions and gestures (the emotions are realized by Baldi's facial expression).

For markup inserted by highlighting text, there are menus for

- specifying the level of emphasis (also for adding or removing preset parameters) and varying the accompanying intonation contour;
- for specifying the values of the Sable SayAs tag's attributes (e.g., normal, spell, syllabify, etc.);
- adjusting the details of pitch range and contour by manipulating a graphic display (only with preset emphasis parameters).

OGIsable supports a number of markup tags; the following description is reproduced from the readme file accompanying the GUI download:

- BREAK
  - LEVEL="LARGE" : this starts a new utterance if LEVEL is set to NONE, a break cannot occur where the tag is inserted. if LEVEL has any other value, a minor phrase break is introduced.
  - MSEC="500" : this inserts a pause of half a second.
- AUDIO
  - SRC="laugh.wav" : this inserts a laugh sound in the synthesized speech other sounds currently available are:
  - `annoyed.wav` `hihi.pk` `relief.wav` `sniff.wav`
  - `surprise.wav` `yawn.wav` `hihi.wav` `shh.wav`
  - `breath.wav` `hmm.wav` `smack.wav` `tape2c.sp`
  - `burp.wav` `laugh.wav` `smack2.wav` `victory.wav`
  - `haha.wav` `release.wav` `sneeze.wav` `uhm.wav`
- PRON
  - SUB="edin bruh" : this replaces the pronunciation of the tagged text
  - IPA="w oU 9r l d b E t" : specify pronunciation using Worldbet which is a computerized version of IPA.
- SAYAS
  - MODE="phone" : pronounce tagged text (i.e. number) as a phone number current modes: phone,digits,literal,cardinal,ordinal, syllabify. The 'literal' mode will spell text or read digits. The 'syllabify' mode enables rhythmic syllabified saying of the tagged word(s).
  - MODETYPE="fluent" : For MODE="literal", this means the text is spelled fluently. The other option is MODETYPE="isolated". For MODE="syllabify", the MODETYPE options are "automatic" or a specification using Worldbet symbols, such as "s I . l @ . b I . f aI". The dots are used to indicate the syllable boundaries.
- SPEAKER
  - NAME="abc" : this changes the voice to abc (Mexican spanish male) OGI voices: mwm, tll, jph, aec, abc, hvs, axk, bcs, ogirab, mwm2tll, mwm2jph, mwm2axk. `http://cslu.cse.ogi.edu/tts/demos` for complete overview.
- RATE
  - SPEED="-20%" : lets speaking rate drop with 20 % you can also specify one of (fastest,fast,medium,slow,slowest).
- PITCH
  - BASE="-10%" : this lets the average pitch drop with 10 % you can also specify one of (highest,high,medium,low,lowest).
  - RANGE="+10%" : this increases the pitch range with 10 % (more dynamic speech) you can also specify one of (largest,large,medium,small,smallest).
  - TOBI="H*;H-H%" : this enables specification of Tobi intonation events. If a separator ";" is found, the first label is assumed to be the pitch accent and the second label is the boundary tone.
  - CONTOUR="0.1 100; 0.3 150; 1.0 100" : allows specification of a pitch contour. The first coordinate is relative time and the second coordinate is pitch in Hertz.
- VOLUME
  - LEVEL="+10%" : increases volume with 10 % you can also specify one of (loudest,loud,default,medium,quiet).
  - CONTOUR="0.1 1; 0.3 1.5; 1.0 2.0" : allows specification of a pitch contour. The first coordinate is relative time and the second coordinate is relative volume.
- MARKER

- EMBED="some text" : this places a marker in the input text. This tag can be used to communicate specific timing information to the engine, e.g. to drive facial movements of an animated face.

5.6 Conclusion

The CSLU Toolkit provides a number of applications for a diverse range of purposes – ranging from the pedagogical to the theoretical – in multimodal human-computer interaction – in particular, in the synthesis of text-to-speech and accompanying facial expressions. Some of the Toolkit's principal components (namely, RAD and BaldiSync) are designed for purposes, such as dialogue modelling, that fall outside of the concerns of ISLE WP11. However, the core SpeechView tool and the auxiliary OGIsable GUI markup tool do have functionality relevant to multimodal corpus annotation (though one of the basic desiderata of annotation tools, namely, search capability, is entirely lacking in the Toolkit).

The capability that SpeechView provides for annotating waveforms (or some aligned representation of a given waveform, such as a spectrogram or a pitch contour) with multiple layers of arbitrary time-aligned labels, gives the user a great deal of flexibility in coding and representing data. However, while it is possible to store, reload and edit specific time-aligned waveform labellings, SpeechView does not offer the possibility of fixing a set of labels, which would function as a consistency check during annotation. Such functionality is provided by the OGIsable GUI, since it contains a repository of graphic symbols and textual display options that are inserted into the annotation by clicking (it is not clear, however, whether the user is strictly bound to this repository or can define extensions to it).

Both SpeechView and the OGIsable GUI have easily functional and comfortable user interfaces; the markup GUI in particular insulates users from having to deal directly with XML code, though those who are so inclined have the option of doing so. Similarly, SpeechView's additional command-line options allow more experienced users to undertake additional fine-tuning (though most of that functionality is not related to labelling, i.e. to annotation).

The most significant limitation of these tools with respect to corpus annotation may be scalability. The functionality they provide is easy to apply to fairly small texts or dialogues, but it seems likely that it would be quite laborious to annotate large corpora with them. This is especially an issue with SpeechView, since it does not support annotation consistency control.

Another issue is portability. SpeechView, as a core component of the Toolkit, is currently confined to the Windows platform, though since it is built on top of the Festival TTS engine, which also runs on various Unix platforms, there appears to be no principled barrier to reimplementing it for such platforms (in fact, Festival was ported to Windows specifically for inclusion in the CSLU Toolkit). This should be even less of an issue for the OGIsable GUI, since it is a Tcl interface for XML-conformant markup and hence essentially platform-independent (it can in fact be used as a stand-alone tool apart from the Toolkit, though then without the TTS and facial expression support).

With respect to standardisation, SpeechView is basically neutral, since it neither enforces nor prohibits any particular type of annotation, though again its lack of support would probably make it difficult or impossible to guarantee conformity to a chosen standard. The OGIsable GUI is better situated in this regard, since it is based on the XML-conformant Sable markup language, which may be considered a de facto standard for TTS annotation; it is not clear, however, whether the incorporated OGI extensions to Sable would be accorded this status.

5.7 References

M. M. Cohen, J. Beskow, and D. W. Massaro. Recent developments in facial animation: An inside view. In *AVSP '98*, Sydney, Australia, December 1998. http://mambo.ucsc.edu/psl/avsp98/11.{ps,doc}.

Ron Cole, Dominic W. Massaro, Jacques de Villiers, Brian Rundle, Khaldoun Shobaki, Johan Wouters, Michael Cohen, Jonas Beskow, Patrick Stone, Pamela Connors, Alice Tarachow, and Daniel Solcher. New tools for interactive speech and language training: Using animated conversational agents in the classrooms of profoundly deaf children. In *Proceedings of ESCA/SOCRATES Workshop on Method and Tool Innovations for Speech Science Education*, London, UK, April 1999.

Pamela Connors, Alice Davis, George Fortier, Kerry Gilley, Brian Rundle, Chris Soland, and Alice Tarachow. Participatory design: Classroom applications and experiences. In *Proceedings of the International Conference of Phonetic Sciences*, San Francisco, CA, August 1999.

R. Sproat, P. Taylor, M. Tanenblatt, and A. Isard. A markuplanguage for text-to-speech synthesis. In *Proceedings of the Fifth European Conference on Speech Communication and Technology (Eurospeech 97)*, Rhodes, Greece, September 1997. http://www.cstr.ed.ac.uk/publications/1997/Sproat_1997_a.ps.

Patrick Stone. Revolutionizing language instruction in oral deaf education. In *Proceedings of the International Conference of Phonetic Sciences*, San Francisco, CA, August 1999.

S. Sutton, R. Cole, J. de Villiers, J Schalkwyk, P. Vermeulen, M Macon, Y Yan, E. Kaiser, B. Rundle, K Shobaki, P. Hosom, A. Kain, J. Wouters, M. Massaro, and M. Cohen. Universal speech tools: the cslu toolkit. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, pages 3221-3224, Sydney, Australia, November 1998. Proceedings of the International Conference on Spoken Language Processing (ICSLP).

Johan Wouters, Brian Rundle, and Michael Macon. Authoring tools for speech synthesis using the sable markup standard. In *Proceedings of Eurospeech*, volume 2, pages 963-966, Budapest, Hungary, September 1999.

6 The MATE Workbench

6.1 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 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 MATE consortium includes NISLab (Odense, Denmark) (coordinating partner), CSELT (Torino, Italy), DFE (Barcelona, Spain), DFKI (Saarbrücken, Germany), HCRC (Edinburgh, UK), ILC (Pisa, Italy), IMS (Stuttgart, Germany), and TID (Madrid, Spain).

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). On this background, the MATE markup framework (Dybkjær et al. 1998, Dybkjær and Bernsen 2000b) was proposed as a standard which can facilitate uniform description of schemes across levels. For each level, 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).

The MATE software workbench (Isard et al. 1998, Isard et al. 2000, Dybkjær and Bernsen 2000a, McKelvie et al. 2000) supports the MATE markup framework and incorporates the MATE best practice coding schemes.

The first release of the MATE workbench appeared in November 1999 and was made available to the +80 members of the MATE Advisory Panel from across the world. Since then, several improved versions have appeared and in May 2000 access to executable versions of the Workbench was made public. The MATE Workbench is now publicly available both in an executable version and as open source software under the GNU LGP license at http://mate.nis.sdu.dk. The Workbench is still being improved by the MATE consortium, so new versions will continue to appear. 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.

The review provided in the following is based on hands-on experience with release version 0.17 from August 2000.

6.2 Software design

The MATE workbench has a modular architecture which facilitates the addition of new modules and new tool functionality by its users. The workbench is implemented in Java. 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.

The MATE workbench has the following major components: An internal database which is an in-memory representation of a set of hyperlinked XML documents; a query language and processor which are used to select parts of this database for subsequent display or processing; a stylesheet language and processor which respectively define and implement a language for describing structural transformations on the database; a display processor which handles the display and editing actions; and a user interface which handles file manipulation and tool invocation.

Data structures and APIs for the MATE workbench are described in (Isard et al. 2000)

6.3 Platform requirements

The MATE 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. Since Java is slow it is recommended to use a fairly fast PC with at least 96 Mb RAM.

6.4 Interface

Starting the MATE workbench will result in the two windows in Figures 6.1 and 6.2 being opened. The control window is the main window from which workbench tool windows can be opened. New tools can be added to the workbench and will then automatically be made available in the tools menu. The 'Help' menu gives access to the online help facility (Figure 6.11). One or more project windows can be opened from the 'File' menu. The project window is used for browsing, adding or editing corpus files. Figure 6.2 shows example projects which come with the workbench and which include best practice coding schemes and annotated examples for a number of different annotation levels.

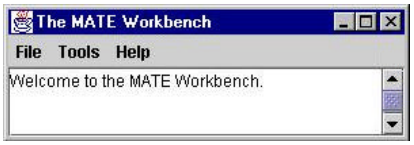


Figure 6.1. The control window.

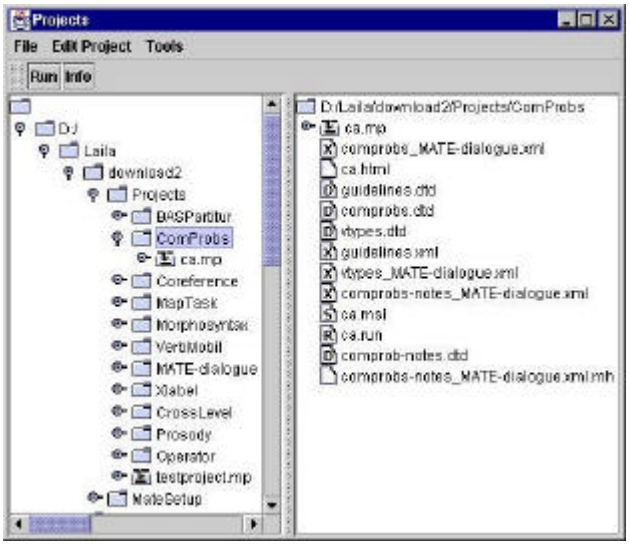


Figure 6.2. The projects window.

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 (Figure 6.3). 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.

Presentation to the user of a dialogue during annotation and of the available tag set depends on the coding scheme being used, cf. the examples in Figures 6.7-6.9. Presentations are based on style sheets. Thus users may very well have to write a new style sheet for a new coding scheme.

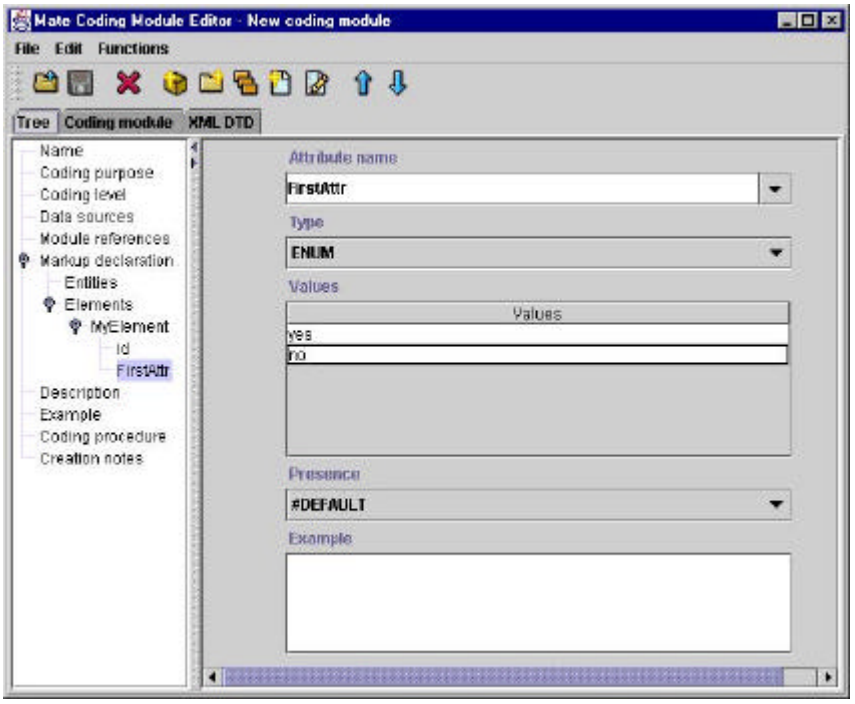


Figure 6.3. The coding module editor.

This is a major usability problem, and we need to find a more 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 6.4. It is high on the 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.

Figure 6.4. Editing a stylesheet.

A second major usability issue is the interface to the query tool and its results. As for the latter, Figure 6.6 makes it evident that usability improvements are needed. The query results could be presented far more transparently using an appropriate style sheet. For example, the document and the annotations used as constraints in the query (plus possibly other annotations) should be displayed with matching instances highlighted. This is also on the MATE to-do list. So is a more comprehensible interface for expressing queries than the present one (Figure 6.5).

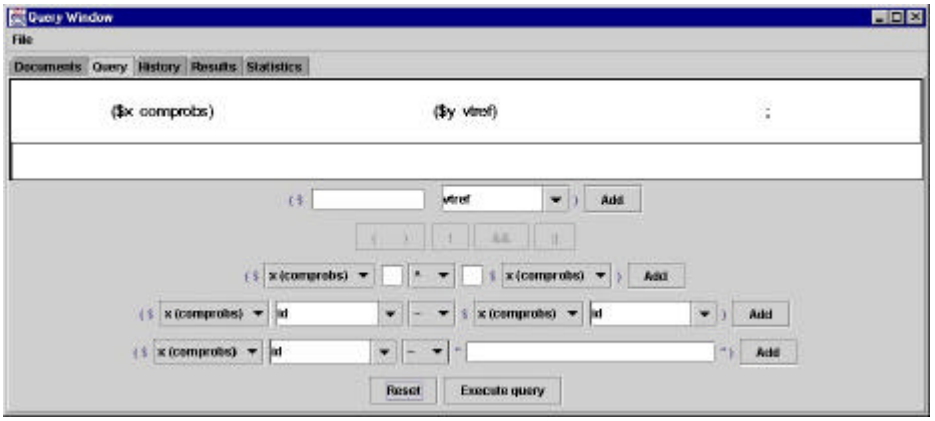


Figure 6.5. The query window.

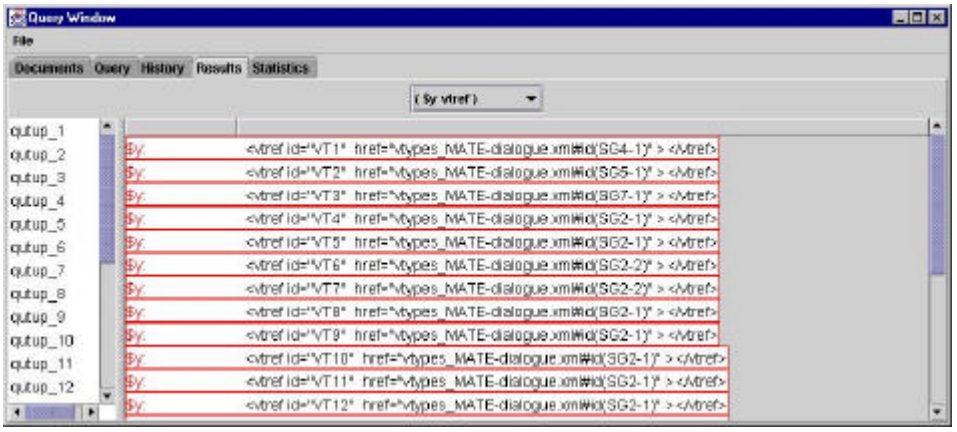


Figure 6.6. Results of a violation types query, cf. Figure 6.8.

Screen shots illustrating the interface during annotation and the help facility are shown in the next section. These parts function well and are easy to use.

6.5 Functionality

The engineering objective of MATE has been to specify and implement a generic annotation tool in support of the MATE markup framework and the selected best practice coding schemes. Several existing annotation tools were reviewed early on to gather input for MATE workbench specification (Isard et al. 1998). Building on this specification, the MATE markup framework and the selected coding schemes, the workbench has been implemented (Isard et al., 2000) including the following major functionalities:

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, cf. the examples in Figures 6.7-6.9. 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.

Clicking on the green 'PLAY' button in Figure 6.7 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.

Figure 6.7. Annotation using the MATE Map Task scheme.

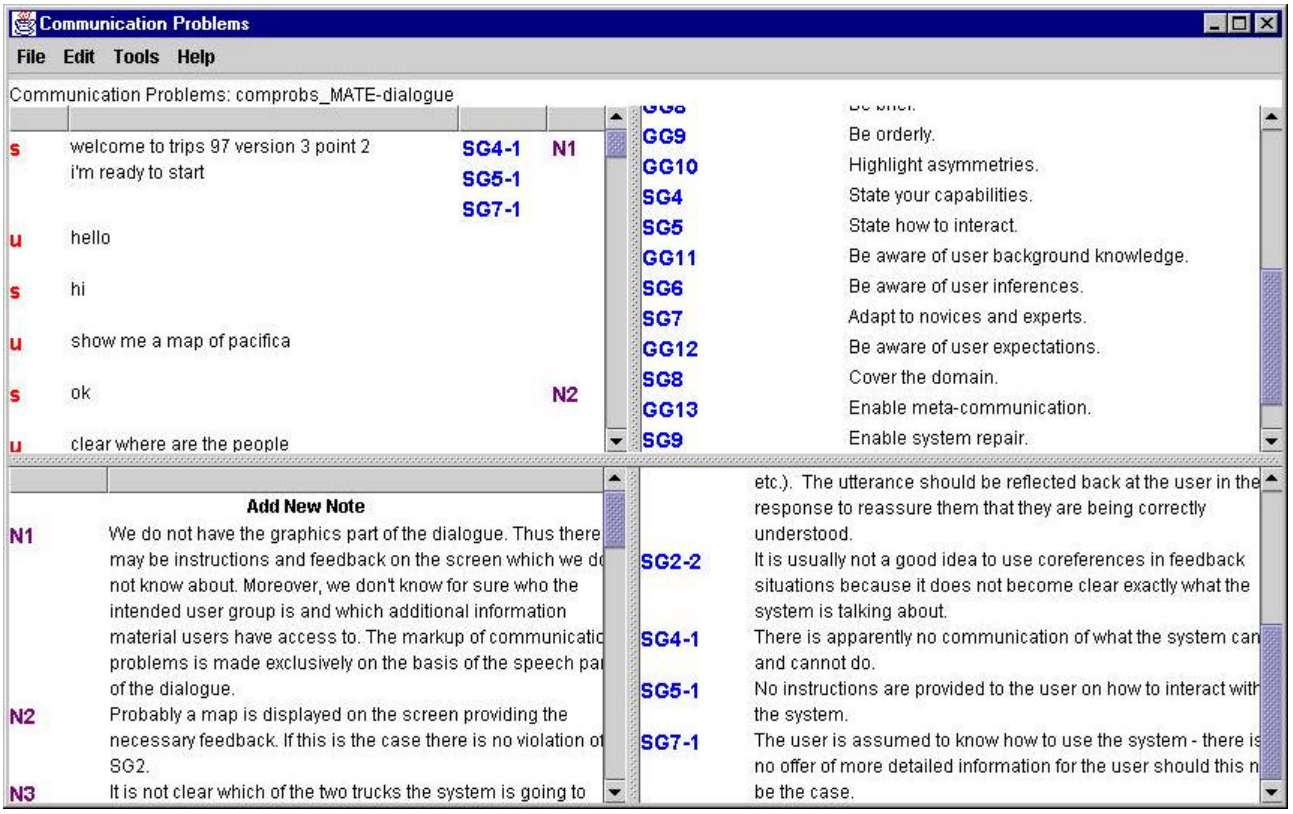


Figure 6.8. Annotation using the MATE communication problems scheme.

In the communication problems window in Figure 6.8 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.

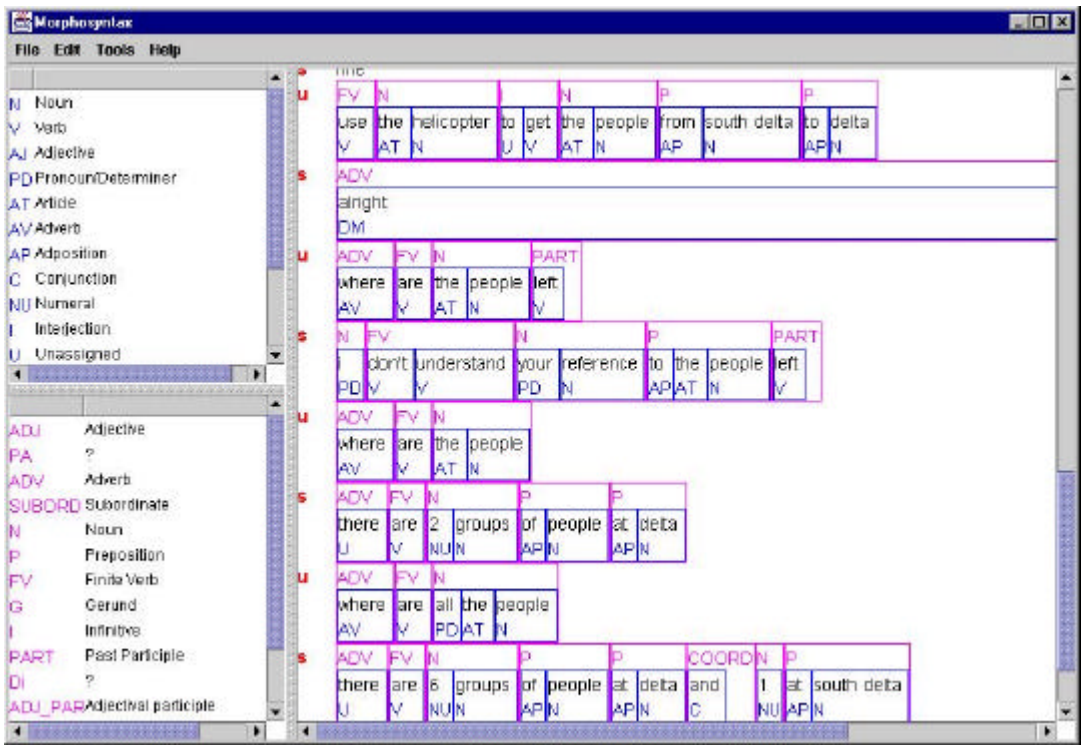


Figure 6.9. Dialogue annotation at the morpho-syntactic level (word level).

In the morpho-syntax window shown in Figure 6.9 the top left-hand pane shows morphological word tags whereas the bottom left-hand pane shows chunk tags. The right-hand pane shows the annotated dialogue. Tags in the dialogue can be changed to new tags selected from the lists to the left.

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.

Users may add new coding modules (coding schemes) themselves for existing or new coding levels via the coding module editor, cf. Figure 6.3. 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 6.3. It prescribes what constitutes a coding, including markup representation and relations to other codings (module references). A full link between the coding module editor and the rest of the workbench is under development at the time of writing.

Audio files can be selected in the projects window (Figure 6.2) and played using the MATE audio tool. The window in Figure 6.10 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.

Import of files from XLabels and BAS Partitur to XML format is supported in order to demonstrate the usefulness of importing widely used annotation formats for further work in the workbench. Other converters can easily be added. Export to file formats other than XML can be achieved by using style sheets. For example, information extracted by the query tool may be exported to HTML to serve as input to a browser.

The workbench enables information extraction of any kind from annotated corpora, cf. Figures 6.5 and 6.6. Query results are shown as sets of references to the queried corpora. Extraction of statistical information from corpora, such as the number of marked-up nouns, is also supported. Computation of important reliability measures, such as kappa values, is enabled.

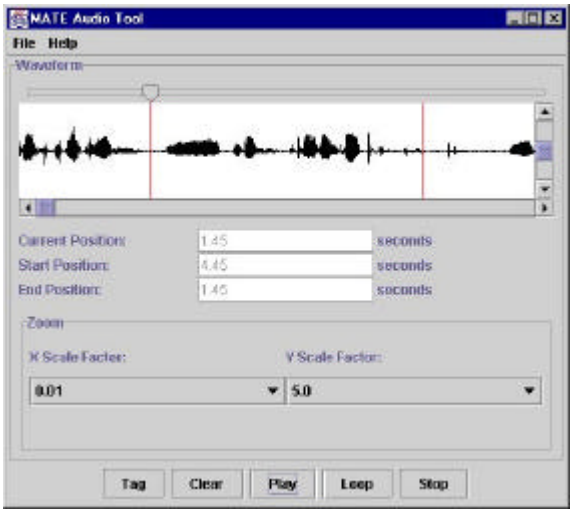


Figure 6.10. The MATE audio tool.

An online help facility may be consulted at any time during use of the MATE workbench. Figure 6.11 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 6.7-6.9.

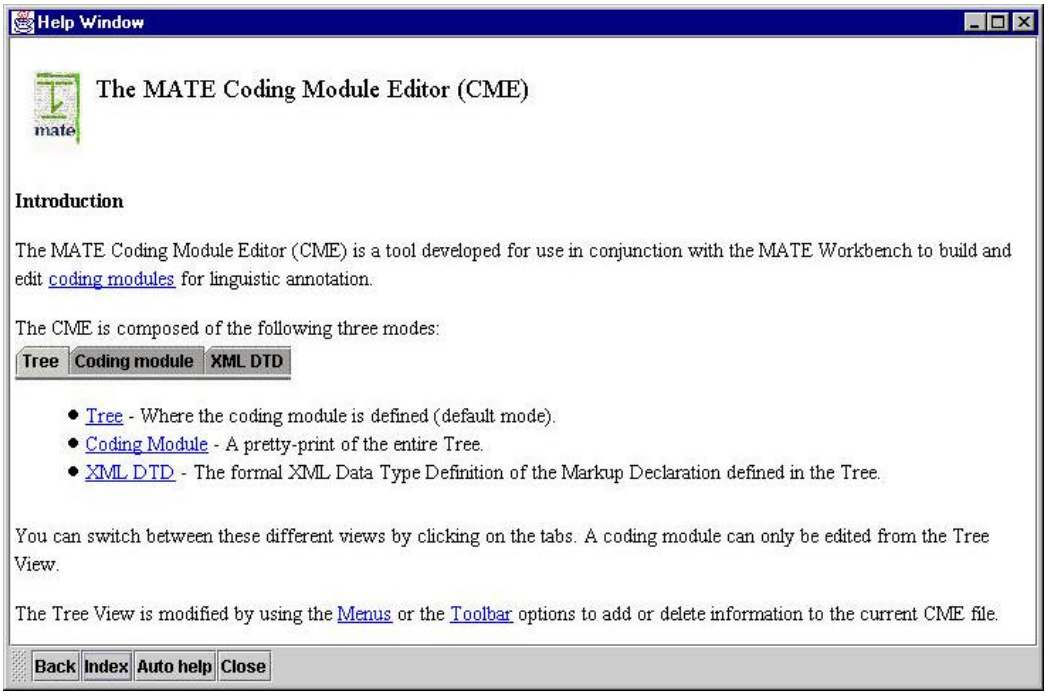


Figure 6.11. The help window.

6.6 Conclusion

MATE has taken a major step towards standardisation and universal support of spoken dialogue data annotation and exploitation by proposing a framework for the annotation of spoken dialogue corpora at multiple levels and by building a workbench in support of this framework. Focus has been on comprehensibility and ease of use.

The MATE workbench still suffers from several shortcomings in particular as regards the user interface, cf. Section 6.4 above, and of course additional functionality will at any time be desirable. The MATE workbench is, however, already being used in a number of projects for annotation.

It is likely that the MATE markup framework is flexible enough to be extended to cover markup of natural interaction and multimodal data. Given its modular architecture, this kind of extension of the

workbench should also be possible. The question is, however, whether it would be recommendable to re-implement at least parts of the workbench to optimise speed and functionality.

6.7 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.

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.

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.

McKelvie, D., Isard, A., Mengel, A., Møller, M.B., Grosse, M. and Klein, M.: The Mate Workbench - An Annotation Tool for XML Coded Speech Corpora. Speech Communication (special issue on Speech Annotation and Corpus Tools), Vol 33, No. 1-2, December 2000.

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.

7 MPI Linguistic Tools: CAVA and EUDICO

7.1 Introduction

CAVA (Computer Assisted Video Analysis) and EUDICO (European Distributed Corpora) are projects (begun in 1994 and 1997, respectively) at the Max Planck Institute for Psycholinguistics (MPI) in Nijmegen devoted to developing tools for the analysis of multimedia (specifically, audiovisual) corpora. The tools are intended to support the scientific exploitation of such corpora by linguists, anthropologists, psychologists and other researchers. The principal tasks of the tools include:

- viewing and manipulating digitized audiovisual files
- multilayer annotation of the data files
- querying of the annotations

The CAVA and EUDICO systems differ with respect to distributed operation. The CAVA tools are platform-dependent, the system employs a proprietary data storage format, and it is designed principally for site-specific use. In contrast, the EUDICO tool suite is intended to be platform-independent, to support various storage formats, and above all to support distributed operation via the internet.

In the original CAVA setup there are two AV transcription and annotation tools: the *Transcription Editor* (TED) and *MediaTagger*. TED transcribes analogue videotape, while MediaTagger transcribes digital AV files. For this review we obtained a copy of MediaTagger, which is distributed as a binary executable file, with an accompanying software manual, by MPI on request free of charge for research purposes.[1] The current version is 3.0b (April 2000). The only information we have on TED is what is contained in the MPI website.[2]

The EUDICO toolbox is still in development and its components are not yet publicly available for inspection. However, the MPI website provides a read-only demonstration version (see http://www.mpi.nl/world/tg/lapp/eudico/eudico.html).[3]

7.2 Software design

7.2.1 CAVA

CAVA is a multi-platform system, incorporating both Macintosh- and PC-based tools, which can access data stored in an Oracle database on a Unix server.

With TED the computer controls the functions of a video tape player and an ASCII text editor is used for annotating the tape by reference to time codes. The MPI website provides no information about TED's implementation.

MediaTagger is an annotation tool for digital AV files in the Apple QuickTime format. Since it is the most extensively developed component of CAVA and a direct precursor to the EUDICO system, and since we had hands-on experience with it, it will be considered here in more detail.

[1] MediaTagger is currently in use at ca. 60 institutes worldwide (P. Wittenburg, p.c.).

[2] See http://www.mpi.nl/world/tg/CAVA/ted/ted.html.

[3] But there appears to be a bug in the access code. We have not been able to run the demonstration. However, we have been able to test it on-site at MPI and have discussed it with the EUDICO project director, P. Wittenburg, and the principal software developer, H. Brugman, to both of whom we are grateful for the invitation to MPI to inspect the tools and for providing valuable feedback for this report. In addition, I would like to thank Peter Wittenburg for written comments on an earlier version. I would also like to thank Ulrich Heid for numerous discussions of and suggestions concerning the content and form of this report.

7.2.1.1 Architecture and Implementation

MediaTagger is the central component of the CAVA system. It supports two modes of operation: as a stand-alone tool for viewing and annotating digital AV files, and as a client to a file or database server. In the latter configuration, the system has a two-layer architecture as shown in Figure 7.1.

The database server has been implemented at MPI under Oracle on a Unix server. Operation in client-server mode allows querying of the transcripts produced by MediaTagger by means of SQL or other proprietary search tools (cf. footnote 9 below).

MediaTagger itself is implemented as a Macintosh application based on QuickTime and requires installation of QuickTime version 2.1 or higher.

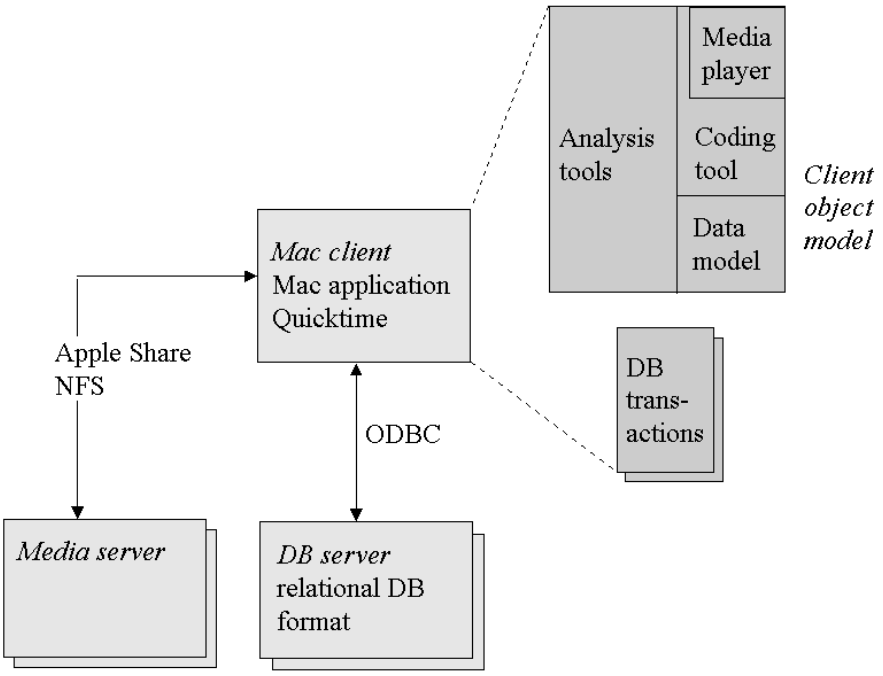


Figure 7.1. CAVA Architecture.

7.2.2 EUDICO

7.2.2.1 Architecture and Implementation

EUDICO's design goals of distributed operation and platform and annotation format -independence dictate a mostly three-layer architecture as shown in Figure 7.2.

As with CAVA, the outer layers of the EUDICO architecture instantiate a client-server model. Format-independence is achieved by adding an intermediate encapsulation layer. This is provided by a Java server, which the client interacts with via Java remote method invocation (RMI) (or another suitable transfer mechanism). This setup permits a common interface between the client and various data resources, such as database and file servers. In addition to this three-layer architecture, EUDICO also supports direct communication between the client and a streaming media server (the lefthand side of Figure 7.2).

The EUDICO architecture is based on a uniform corpus representation model developed at MPI, called the Abstract Corpus Model (ACM), which is implemented as a set of Java classes (Brugman et al. 2000). This model is intended to embody the basic functions of most linguistic corpora, with special emphasis on annotation types. The tool abstraction layer of the ACM provides the client with a uniform toolset, irrespective of differences in server-side implementation, for all of the corpora supported by EUDICO. The corpus abstraction layer incorporates the minimal tool implementation that the server must provide the client for each corpus resource.

7.2.2.2 Software requirements

The client-side tools of EUDICO require the Java Media Framework JMF1.1 with proper performance pack (Windows or Solaris). To run EUDICO on a browser-independent applet viewer, the appropriate Java Development Kit JDK1.2 is required. To run EUDICO via an internet browser, either the Internet Explorer browser (version 3.02 or higher), or Netscape Navigator (3.0 or higher) with the Java Plugin 1.2 is required.

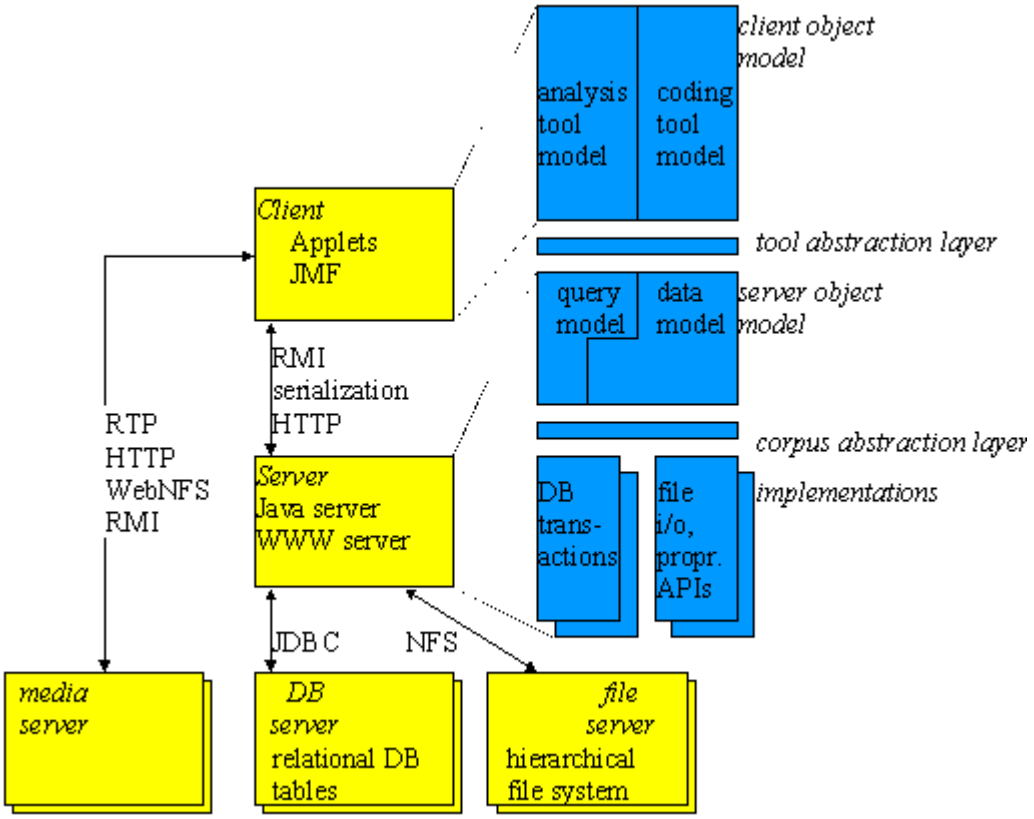


Figure 7.2. EUDICO Architecture.

7.3 Platform requirements

7.3.1 CAVA

7.3.1.1 TED

The MPI website provides no specifics about the computer hardware required by TED, other than that is it PC-based (Windows). But since it is used for transcribing video tape, a video tape recorder and a serial line connecting the computer and the recorder are required.

7.3.1.2 MediaTagger

MediaTagger requires a Macintosh PowerPC with at least 16 MB of memory and sufficient hard drive space to store QuickTime movies (in stand-alone operation).

7.3.2 EUDICO

Since EUDICO runs under the JMF, which requires JDK1.2, it currently runs only on PCs or Sparc workstations (Macintosh does not support the required Java software; however, since JMF is now supported by Linux,[4] EUDICO will soon be ported to Linux as well (P. Wittenburg, p.c.)).

With a PC, the hardware requirements are a Pentium II processor with a speed of 400 MHz, 64 MB memory, and an internet connection with a minimal rate of 400 kbit/sec. With a Sun workstation the requirements are: Sparc Ultra-1 or better; color, preferably 24 bit color; an internet connection of 400 kbit/sec or better.

7.4 Interface

7.4.1 CAVA

7.4.1.1 TED

TED's user interface consists of a split PC screen (Figure 7.3). The upper panel is a text editor and the lower panel is a time-coded transcription list.

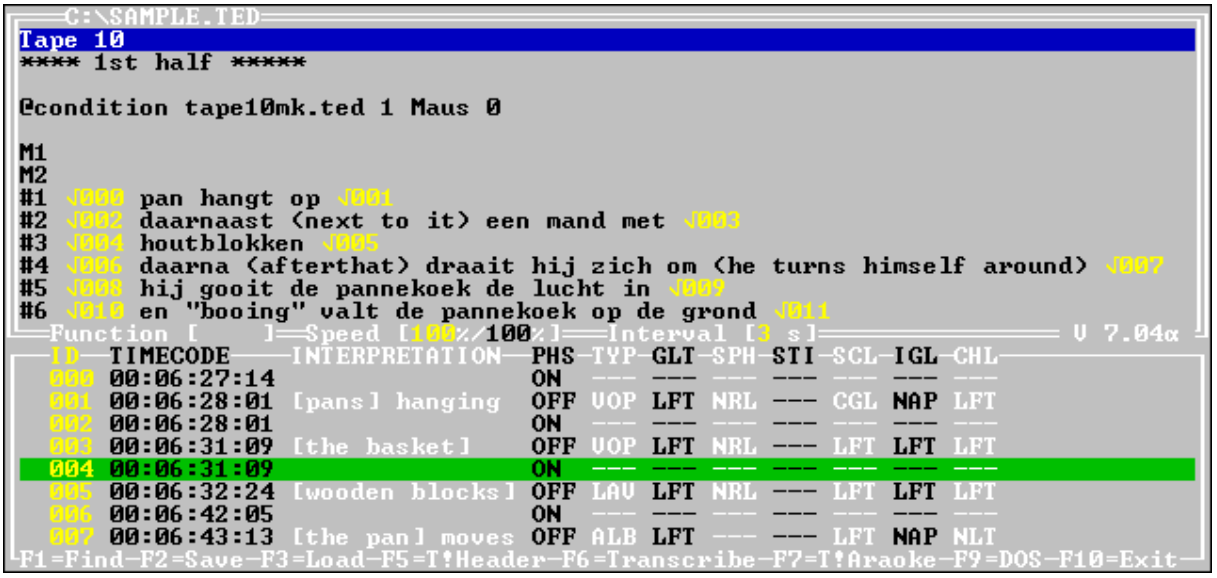


Figure 7.3. TED User Interface.

7.4.1.2 MediaTagger

The startup MediaTagger user interface consists of a typical Macintosh command list display of pull-down menus configured to MediaTagger's functionality (File, Edit, Format), and augmented with additional menus specific to MediaTagger's functionality (Tiers, Tags, Windows). When a video file is opened, two windows are displayed, a media viewer (Figure 7.4) and a listing of annotation tiers (Figure 7.5).

Below the video display, optional subtitles display the time code of the current frame (running when the video is playing) and, for any subset of the annotation tiers, the tags that are coded to the current video segment (the only tag shown in Figure 7.4 is "A"). The buttons beneath the video display control basic functions such as playing and pausing, moving frame by frame, and audio volume. Additional functionality is provided by the button box, from bottom to top as follows: a pull-down menu for activating one of the tiers; one for selecting playback rate; direction of playback; locking the

[4] At least in beta implementation; see http://www.blackdown.org/java-linux/jdk1.2-status/jmf-status.html.

playback to the current segment; moving to the current segment's first or last frame; and moving to the first frame or the previous or next tag on the active tier.

The tier listings window contains a pane with a check-box listing of each of the user-defined annotation tiers; checking a tier enables its display in lower panes of the window. Each displayed tier pane contains the name of the tier and a scrollable list of all its annotation tags, keyed to the starting and stopping time codes of its video segment (i.e, the time codes for the first and last frames of the segment). By clicking on a tag line, that segment of the video is immediately accessed and displayed in the viewer.

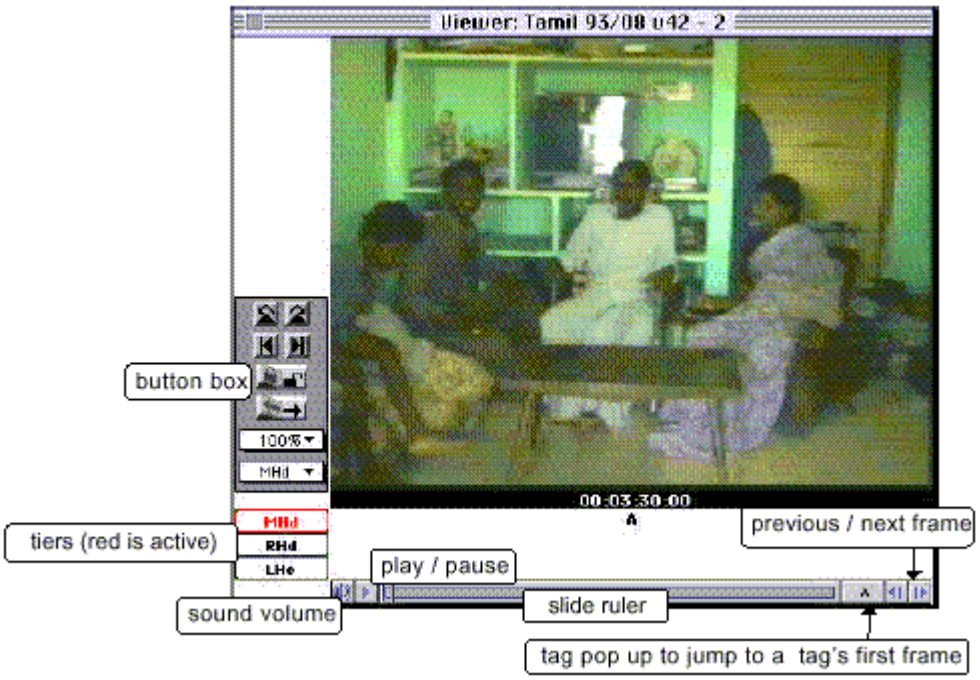


Figure 7.4. MediaTagger Viewer.

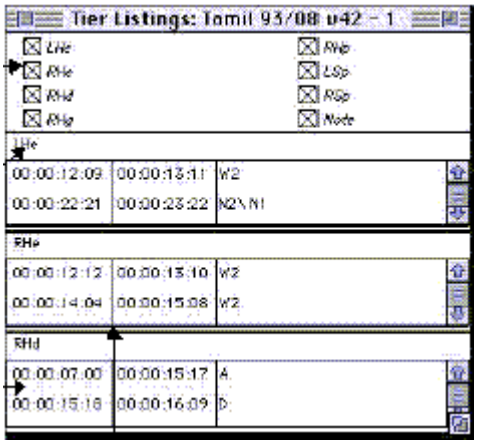


Figure 7.5. MediaTagger Tier Listings Window.

7.4.2 EUDICO

EUDICO runs either over a web browser with the Java Plugin 1.2 or an applet on the client machine. Users are required to log in; this permits access control in the distributed operation of EUDICO. After

logging in a window opens containing hierarchical listings of available corpora[5] and the tools associated with them (Figure 7.6).

In the demonstration version, only the viewers are functional.[6] In Figure 7.7 is a screen shot of media synchronized viewers:

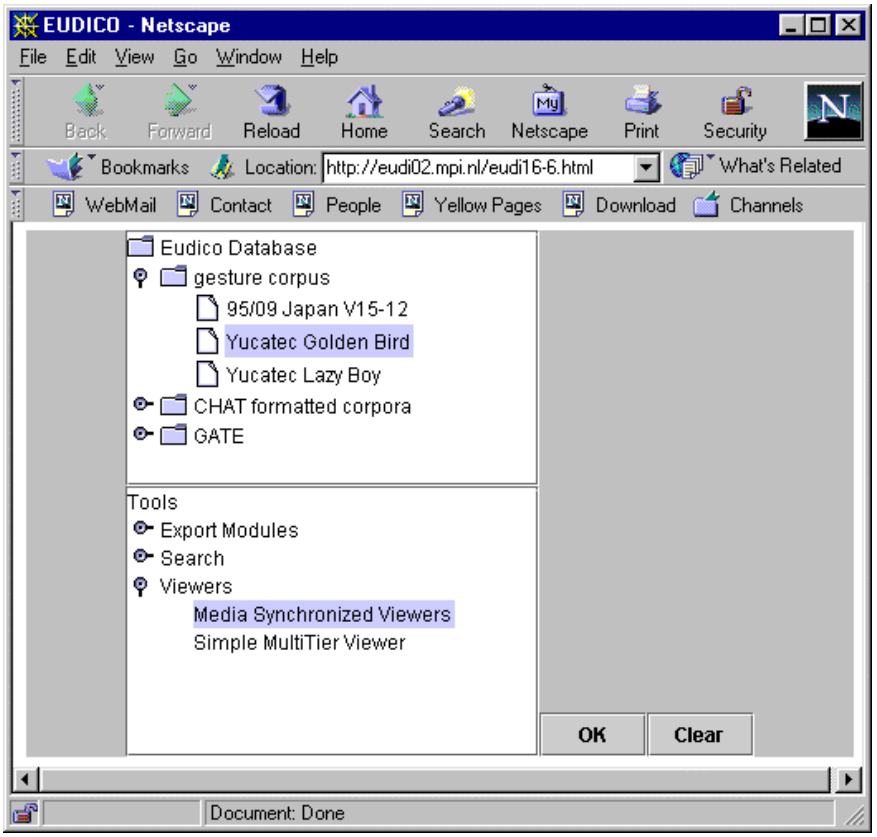


Figure 7.6. EUDICO Browser.

In this viewer an MPEG-1 video is displayed in synchronization with two subtitled annotation tiers[7] and a running time code. The viewer employs the standard JMF audiovisual controls, and a button on the lower right to play the time segment currently selected by clicking on the tier display.

EUDICO also implements several other ways of displaying tags: as a time-ordered list of tag labels and their values (Figure 7.8); as a grid (Figure 7.9), where tag labels head the columns of the display and their values are arrayed below, again time-ordered and scrollable; and on a timeline (Figure 7.10), displaying the duration of the tags. Whereas the first two displays present tag labels and their values for single tiers, the timeline view displays multiple tiers, with tags marking off intervals. The current time of the selected segment is indicated in the timeline viewer by the vertical crosshair.

The layout of the tiers can be modified in various ways: in the subtitle and timeline tag displays, tiers can be reordered by dragging with the mouse. In the grid display, column resizing is possible to accommodate more tags. In the timeline display, a tag's complete value is shown by clicking on its time bar.

---

[5] While this is currently limited to a small sample, it is planned to replace this by MPI's Browsable Corpus, which supports querying metadata; see (Broeder et al. 2000) and http://www.mpi.nl/world/tg/lapp/lapp.html.

[6] A first version of the search function has now been implemented (P. Wittenburg, p.c.), but we have not had an opportunity to see it in operation.

[7] The tiers and their annotation tags are user defined; in Figure 7.7 the tiers display annotated orthographic transcriptions of the recorded speech and descriptions of accompanying gestures ((Brugman et al. 2000) and P. Wittenburg, p.c.).

7.5 Functionality

7.5.1 CAVA

7.5.1.1 TED

"The Transcription-Editor (TED) is designed to make timecode-based transcriptions of video tapes. It gives you the possibility to store various types of (self-defined) information to any timepoint of the tape. All video functions (slow motion, rewind, frame by frame, ...) are remotely controlled by the keyboard to make the work with the video as effective as possible. The output of the TED-program can be stored either in normal text files or in a special DIF file format (transcription-text and timecode-lists)."[8]

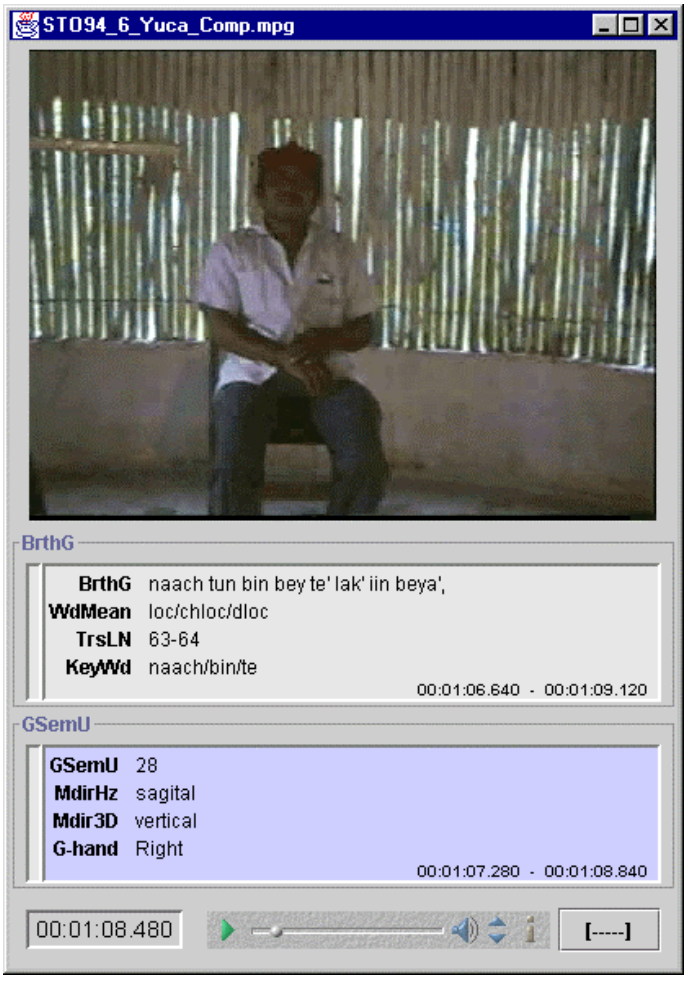


Figure 7.7. EUDICO Media Viewer and Subtitled Tiers.

[8] This and the next citation are from http://www.mpi.nl/world/tg/CAVA/ted/ted.html.

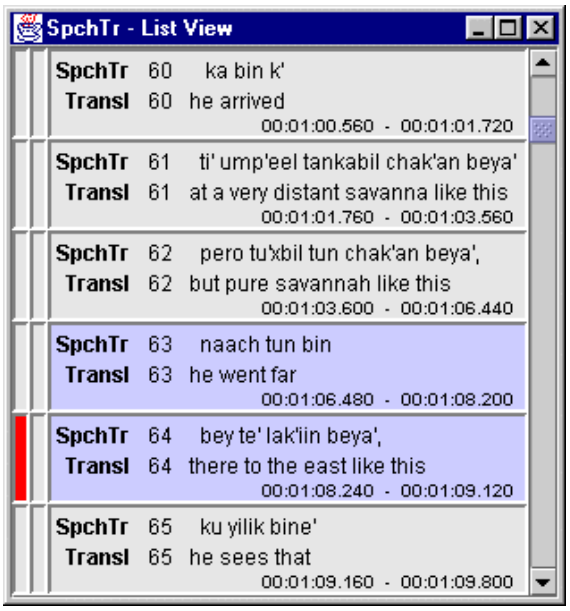


Figure 7.8. EUDICO Taglist Annotation Viewer.

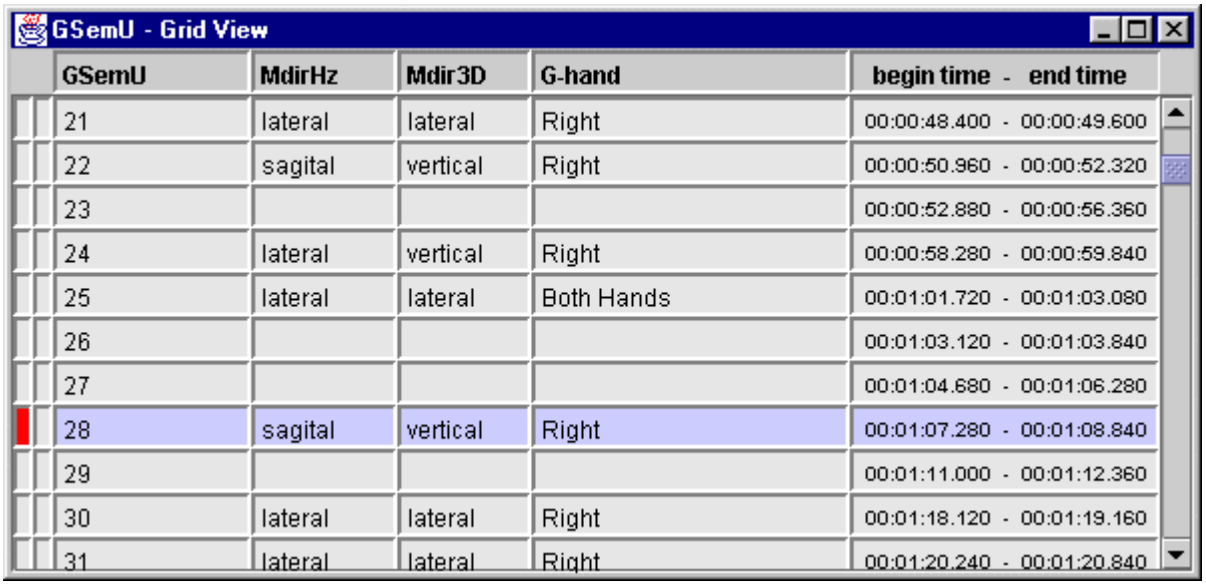


Figure 7.9. EUDICO Grid Annotation Viewer.

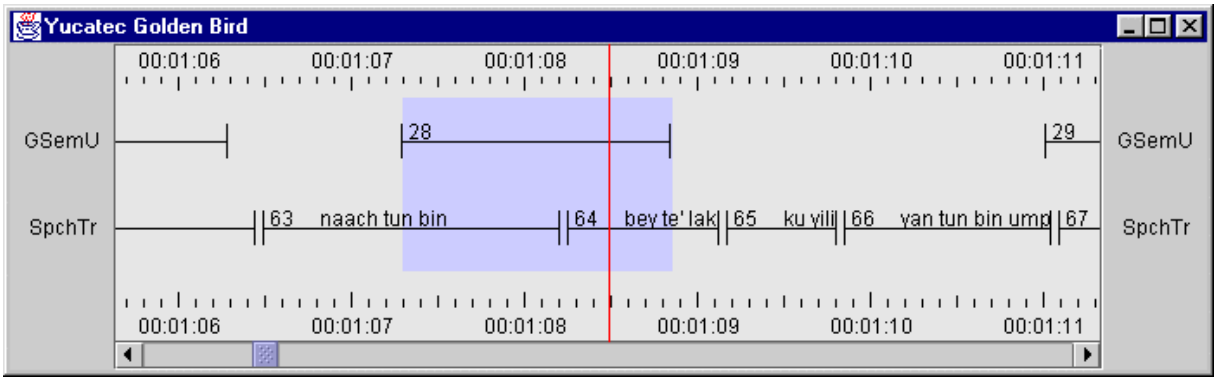


Figure 7.10. EUDICO Timeline Annotation Viewer.

TED's ASCII text editor "is intended for free-form coding or transcription purposes…. [I]t is very easy to take over the real timecode from the video recording to the transcript - it is a matter of pressing

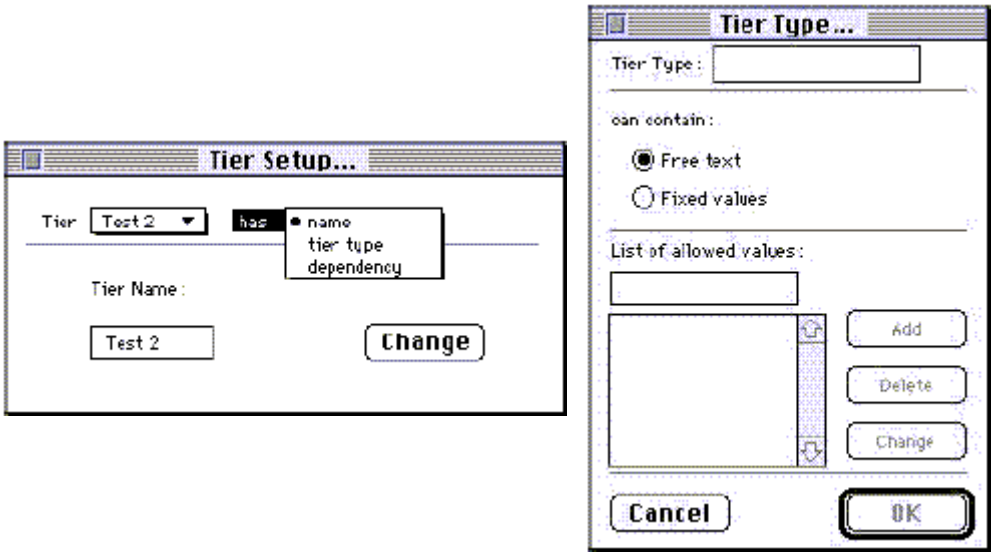
a key. Another efficient option is the possibility to use pre-defined shortcuts to categorize the video material." The website says that TED has many other functions but does not specify them.

7.5.1.2 MediaTagger

MediaTagger has a number of functions for creating annotation tiers and tags for a video transcription. All functions are activated by clicking menu-driven dialogue boxes or pressing key combinations.

Tiers are time-line groupings of tags according to common properties. Each tier may have three attributes: name, type and dependency (Figure 7.11 (a)).

Names are limited to six characters. The type is intended to convey the descriptive or analytic level of the annotations, which may be user-defined, per type, as consisting of free text, fixed textual values, or pairs of numbers specifying two-dimensional screen coordinates.[9] If fixed type is selected, the values are entered by the user and are automatically inserted into a list in alphabetical order (Figure 7.11 (b)).



(a) (b)

Figure 7.11. MediaTagger Tier Setup (a) and Tier Type (b).

The dependency attribute specifies the temporal relations between tiers as determined by the temporal spread of the tags in each tier. Three possibilities are admitted: no dependency; temporal inclusion, for when the temporal extent of the tags of one tier is properly included in that of another (the parent) tier; and simultaneity, which is used for providing distinct annotations that are temporally coextensive. To avoid conflicting time restrictions between dependencies, a given tier can have only one parent tier. The Tiers menu has options for adding and deleting tiers (dependent tiers are also deleted); editing tier attributes; synchronizing MediaTagger and video time codes; and displaying tags as subtitles.

Tags are inserted on the active tier by selecting a video segment. A new tag may be added or the current tag may be deleted (along with dependent tags) or its time codes may be edited (the time codes of dependent tags are then automatically accommodated).

[9] The last type is new to version 3.0b; it is listed in the readme file that comes with the MediaTagger distribution, but is not documented in the accompanying software manual, which has not been updated since version 2.01. The screen shots included in this review are taken from the software manual, hence the absence of the screen coordinate tier type in Figure 7.11(b).

It is also possible to do a simple string search over the tags defined on the currently selected tier (only, i.e., dependent tiers are not included in the search). When a match is found, the media viewer jumps to the first frame of the video segment whose tag contains the matched string.[10]

For adding new tags, a window is opened which is time-coded to the selected video segment (Figure 7.12). This contains, in accordance with the user-defined tier type, either a text box or a clickable pop-up menu from which a user-defined fixed value is selected. The text box contains either free text typed in by the user or screen coordinates, which are automatically inserted by clicking on a location in the media viewer video frame.[11]

Another function is the option of automatically and sequentially adding the contents of an imported text file to an existing tag in punctuation-separated strings; this can be used to align a text transcript to the video display. The aligned text is annotated only to the selected tag, not to the video segment per se.

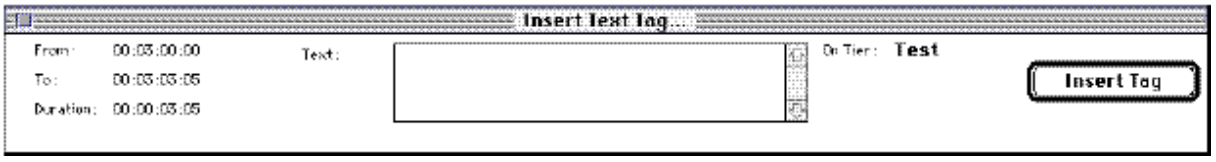


Figure 7.12. MediaTagger New Tag Insertion.

It is also possible to export the textual contents of all the annotated tiers to a file; the contents are arranged in columns providing the beginning and ending time codes and frame numbers of each video segment, the name of its tier, and its tag. This format facilitates manipulation of the annotations by text-processing or spread-sheet programs.

A further useful function is the possibility of opening new (including multiple) viewers for any video segment, which facilitates comparison. These segment viewers lack the full functionality of the main viewer, but permit playing, pausing and volume control. In addition, selected video segments, together with their tag subtitles, may be saved as separate files, also in non-Macintosh format (but then without the tags).

7.5.2 EUDICO

The functionality of EUDICO is currently limited by its read-only implementation. Aside from modifying the layout of the various viewers, mentioned above, the current functionality centres on coordinating the displays. When a media viewer is selected to view a video file, the user may select any or all of the annotation viewers for display with the video (Figure 7.13). All selected viewers are synchronized to the video display and each other, and the displayed tags follow the course of the video.

[10] When MediaTagger is used in the CAVA client-server setup (under Oracle), it has access to more powerful search tools developed at MPI. A search tool with a command line interface allows regular expression querying over any block-structured text formats (in particular, the CHAT and Shoebox formats are supported). An example of its capability is shown at http://www.mpi.nl/world/tg/lapp/searchtool/searchtool.html. A prototype of a graphical Query Generator for CAVA, announced on the MPI website (see http://www.mpi.nl/world/tg/CAVA/mt/Query_gen.html), is now in operation at MPI (P. Wittenburg, p.c.). It has a graphical interface that displays temporal relations. Among its other functionalities, clicking on query results causes MediaTagger to play the corresponding video segment.

[11] Only single pairs of screen coordinates are permitted per video frame.

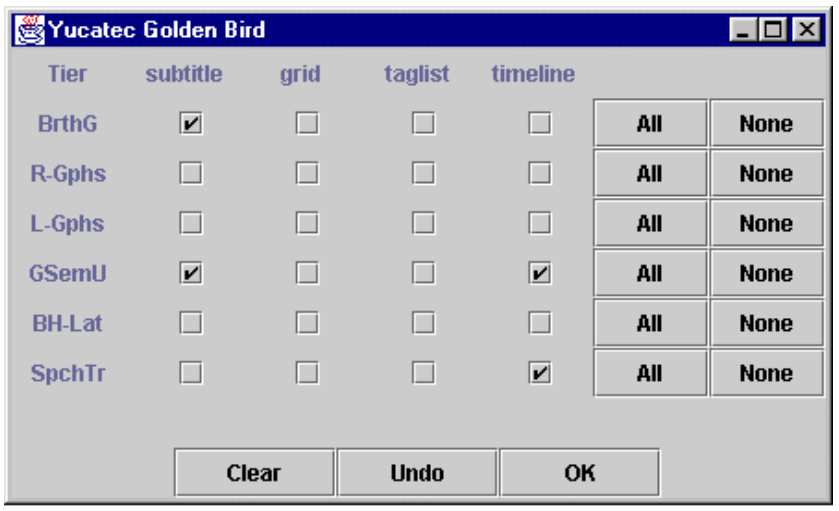


Figure 7.13. EUDICO Viewer Selection Window.

In the video display, the time code can be entered or changed manually. A video segment can be selected by clicking on any displayed tag.

When the first release version of EUDICO is available, it should contain most of MediaTagger's functions as well as new functionality (Brugman et al. 2000). This includes tools for time-coding and then tagging video or audio segments, with typed values supported, to facilitate consistency, and constraints between tiers definable. In addition, as noted above, a first version of a search tool for EUDICO has been implemented that supports, for example, querying combinations of tags according to specified temporal relations, and also statistical queries about the corpus, such as the number of occurrences of certain tags. Another extension that is planned is support of corpus browsing via metadata; this is currently being pursued at MPI with the development of the Browsable Corpus (Broeder et al. 2000), which will augment the current EUDICO corpus selection interface (cf. footnote 5).

In terms of EUDICO's client-server functionality, support is planned for four modes of operation (Brugman et al. 2000): (i) remote media and data servers, (ii) local media server and remote data server, (iii) remote media server and local data server, (iv) local media and data servers. The first mode realizes true distributed operation, both for accessing and annotating corpora. The second mode would be useful for institutions that lack hardware support for high-bandwidth streaming media, but can access remote annotation data. Modes (i) and (ii) require supporting concurrent versions, such as simultaneous annotation of the same corpus; such support is planned. Mode (iii) permits local control of annotations for privacy or security reasons. Mode (iv) effectively instantiates a stand-alone workbench, and could be used by field researchers.

7.6 Conclusion

MPI's CAVA and EUDICO toolkits constitute quite general and sophisticated annotation and analysis platforms for multimodal – specifically, audiovisual – corpora, and hence are directly relevant to the concerns of ISLE WP11. Since EUDICO is essentially the successor of CAVA, it has drawn on and extended its design and functionality, incorporating its advantages while overcoming some of its limitations – at least, that is the goal, since EUDICO is still in development.

In terms of architecture, CAVA implements a two-layer client-server model, which limits accessibility to those users who have the required client-side (Macintosh for MediaTagger or PC for TED) and server-side (Oracle database) setups. EUDICO improves on this architecture by adding an intermediate Java server as an encapsulation layer, which presents users with a uniform client-side interface and uniform services, based MPI's Abstract Corpus Model. This set-up effectively realizes format-independence and it is also platform-independent to the extent that the hardware supports the necessary Java programs (not all platforms currently do so). In this regard EUDICO supports ISLE's standardisation goals. Further support will be achieved when EUDICO incorporates MPI's Browsable Corpus (BC) for querying over metadata, since BC is based on an internal XML representation.

With regard to multimodal annotation functionality, CAVA's MediaTagger has a number of desirable features, and also some limitations. The advantages include arbitrary user-defined annotation layers (tiers), which are typed and can be made dependent on other tiers; and arbitrary user-defined annotation labels (tags) for each tier, which are time-aligned to selected segments of the digital video film being annotated. While tags may be freely inserted, basic consistency control is supported by allowing the user to define a fixed set of tags for given tiers, which are then inserted by clicking (mixing of free and fixed tags in a tier is not possible, though the fixed set can be modified at any time). MediaTagger provides some functionality for aligning arbitrary text with video segments, though this is not very sophisticated, being restricted to single tiers, regardless of dependency relations. Perhaps the most significant limitation of MediaTagger is with respect to querying of annotations; only simple string searches over the tags of a given tier (disregarding dependencies) are directly supported. In client-server operation, MediaTagger can receive input from more sophisticated query processors, but this functionality is bound to CAVA's resource-dependent client-server architecture.

EUDICO does not yet incorporate all of MediaTagger's annotation functionality, though it is planned to implement most of the features. However, EUDICO already has several additional desirable capabilities, chiefly in terms of displaying annotations. Perhaps the most interesting component here is the Timeline Viewer, which shows the timeline relations of the tags in all (or selected) annotation tiers. Also of interest is the complete synchronisation of all annotation viewers with the video displays, including jumping to selected segments. As noted above, the implementation of a sophisticated integrated query tool for EUDICO has been achieved in a first version, which promises to substantially increase EUDICO's utility as an annotation and analysis tool, especially when this functionality is augmented with the metadata capacity of the Browsable Corpus.

7.7 References

Broeder, D.G., Brugman, H., Russel, A., and Wittenburg, P. A Browsable Corpus: accessing linguistic resources the easy way. LREC 2000 Workshop, Athens. Available online at http://www.mpi.nl/world/ISLE/documents/papers/broeder_paper.pdf.

Brugman, H., Russel, A., Broeder, D., and Wittenburg, P. EUDICO, Annotation and Exploitation of Multi Media Corpora. LREC 2000 Workshop, Athens. Available online at http://www.mpi.nl/world/ISLE/documents/papers/anno-Workshop-after.pdf

CAVA website: http://www.mpi.nl/world/tg/CAVA/CAVA.html

EUDICO website: http://www.mpi.nl/world/tg/lapp/eudico/eudico.html

8 Multitool

8.1 Introduction

Multitool is the result of the project: A Platform for Multimodal Spoken Language Corpora (http://www.ling.gu.se/gsmlc/) which was part of: the Swedish Dialogue Systems project (http://www.ida.liu.se/~nlplab/sds/index.html).

The Swedish Dialogue Systems project consisted of a pilot project and a larger, full project.

The aim of the pilot project was to analyse problems connected to building a tool like Multitool and surveying other existing tools which to some extent had the same features as the planned Multitool was to have.

The aim of the full project was to build a platform, i.e. a set of tools, in support of the establishment and use of multimodal spoken language corpora, and to create such a corpus by adapting the existing Gothenburg Spoken Language Corpus (http://www.ling.gu.se/multitool/related.html) to this platform. The resulting platform is Multitool. Multitool is unstable and has several problems, but the project is finished and no funding is available to improve the tool at the moment.

The targeted users of the corpus and tool set are linguistics researchers both from university and industry who wish to analyse spoken language in all its facets, ranging from visual to statistical. Results from such research can be used in the development of practical applications such as multimodal human-machine interfaces in industry, education and entertainment.

The tools in the platform include:

1. a tool for digitising audiovisual corpus data;
2. a tool for synchronous alignment of audio/video and transcribed text;
3. a multimodal corpus presentation tool accessible from the Internet. Audio/video presentation with simultaneous scrolling in a partiture-formatted transcription with highlighting of the phrase being spoken;
4. a transcription coding tool that includes simple corpus presentation using standard and/or partiture format, with optional use of an audio/video presentation extension;
5. a statistics tool able to process information from the coding tool and the corpus.

The project has made use of existing tools for digitisation and statistics. The project has thus been concerned with developing tools 2, 3 and 4.

Multitool is an attempt to build a general tool for linguistic annotation and transcription of dialogues, browsing, searching and counting. An objective of the tool is that it should be able to handle any number of participants, overlapping speech, hierarchical coding schemes, discontinuous coding intervals, relations and synchronization between codings and the media file.

Multitool currently exists in a beta version. One needs Java1.2 and at least Java Media Framework JMF1.1 to make the tool work. The current version of all files can be downloaded from http://www.ling.gu.se/SDS/multitool. A password is needed to go there, and can be obtained by contacting Leif Grönqvist (leifg@ling.gu.se). The tool in its current version is freely available, including the source code.

The links to JDK and JMF were not working on the web page from which Multitool can be downloaded (http://www.ling.gu.se/multitool/). But we managed to find this software anyway. However, when all files where finally obtained, it was not possible to make Multitool run properly. We came as far as being able to open some of the windows but whenever we tried to do something in those windows, the tool broke down immediately. Therefore it has not been possible to get hands-on experience with the tool, so the review is based on descriptions partially obtained through contact with one of the developers.

8.2 Software design and platform requirements

The implementation language is Java and JMF, which should make the tool platform independent. At the moment Multitool only runs on Windows - minimum requirements are 200MHz, 64MB. When

Java JDK and JMF work on the other platforms (Mac/Linux/Solaris), it is expected that Multitool could be ported without too much effort.

When Multitool is running there is one instance of the class MTDoc, and some (zero, one or many) instances of each view type.

MTDoc contains two types of data - codings and synchronizations - built up by the following basic objects:

Timepoint (TP)	The time in seconds from the beginning of the mediafile.
Codingpoint (CP)	A positive integer representing the moment when something starts or ends. A lower value means that it happens before, but it doesn't say anything about the times in seconds.
String	A text string.
Segment	A pair of CP:s - one for the start and one for the end.
Interval	A list of non overlapping segments. If the list consists of more than one segment, the interval is discontinuous.

A coding is a pair of intervals (I1, I2), a (possibly empty) list of participants (P), and a string (V). It should be interpreted as a relation between I1 and I2, associated to P. V is the hierarchical value where the first level is called coding scheme. In many cases I2 is empty and I1 continuous. A synchronisation consists of a TP and a CP and says that the CP happens at the time in the TP. Some examples of codings and synchronisations are shown below:

I1	I2	P	V
31-32		S	comment·gesture·nods
31-32		S	text·mm
32-33		S	text·you {a}re right
33-34		S	text·about that
33-34		A	text·have you
33-35	35-36	A,S	question-answer·yes/no
34-35		A	text·the new movie
35-36		S	text·yes
35-36		S	parts_of_speech·fb
36-37		S	text·i have

CP	TP
31	22.08
33	23.10
36	23.96

The codings and synchronisations are stored in two tab-separated textfiles. The utterances (as shown in the standard view) in this case are:

S: mm you {a}re right about that
A: have you seen the new movie
S: yes i have

And the corresponding partiture:

S:	mm	you {a}re right about that			yes i have	
S:	gesture·nods					
S:						
A:			have you	seen the new movie		
A:						
A:			yes/no			
CP:	31	32	33	34	35	36
TP:	22.08		23.10			23.96

8.3 Interface and functionalities

Clicking on a bat-file which is included in the download file of Multitool, starts the tool. The black DOS-window can be changed to an icon if one wishes to do so.

The main window of Multitool is shown in Figure 8.1. This window appears when the tool is started. The file menu contains the normal items for a file menu. However, at the moment one cannot start annotating from scratch on a new video/audio file – only files that have been designed for Multitool can be used. The user should use SAVE to save all changes made rather often since Multitool is not very stable. The VIEWS-menu entries may be used to create new views of different types. One may quit Multitool via the FILE menu or with the CLOSE-button in the top right corner and all changes will be saved automatically (including codings, information on which views were open, their size and position – basically the program saves the user's last position in the program).

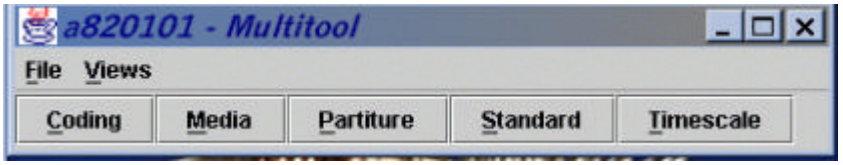


Figure 8.1. The main window.

In the current version of Multitool the coding view, cf. Figure 8.2, shows a tree view of the codings and the number of occurrences of each coding value.

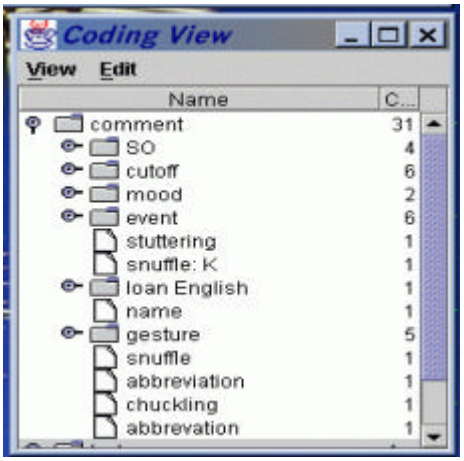


Figure 8.2. The coding view.

The Partiture View, cf. Figure 8.3, has one line for each participant and the codings (utterances) are viewed in chronological order along the x-axis. This gives a clear view of the dialogue structure and the overlapping sections. The lines marked with CP and TP show coding points and time points. It is possible to edit codings and synchronisations in this view. A double click on a speaker line will make a dialogue box appear – either NEW CODING or EDIT CODING – which makes it possible to add a coding or edit the scheme, value and coding point interval. A coding point consists of two logical time

points in a coding. A double click on the time point row will invoke a NEW/EDIT DIALOGUE for synchronisations and a double click on the coding point adds a new coding point.

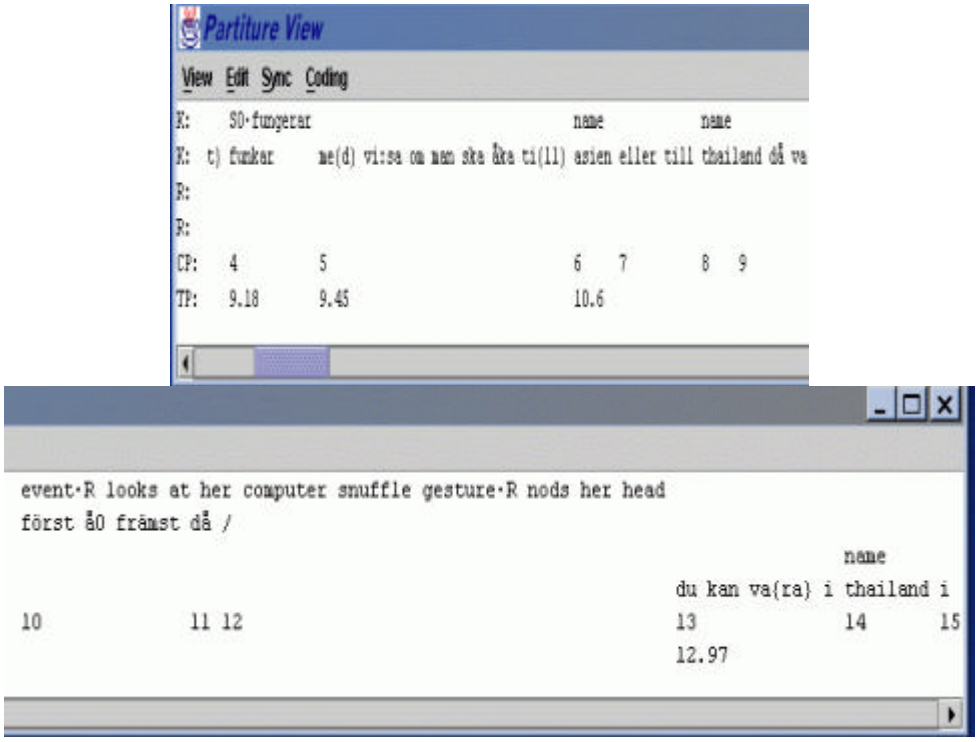


Figure 8.3. The partiture view.

The Media View is meant for playing audio and video, cf. Figure 8.4. The user can navigate through the media file to find interesting sections. It can handle up to fifty different formats, among these: AIFF, AVI, MPEG1-2-3, QuickTime, Sun Audio and Wave. The slider at the bottom may be used to control the movie. The numbers in the text field show the time in the movie for the last start, stop or set time event followed by the times when the user last clicked on the SET A and SET B buttons and the movie speed. If the user clicks the SYNC button, the tool will try to synchronize the other existing views to the media position. In the current version of Multitool there is a problem with different sizes of the Media View. If the time-slider disappears, one should just resize the window until it is shown again.

The Standard View of the Media View, cf. Figure 8.5, shows one utterance on each line. SYNCRONIZE VIEWS in the SYNC menu will cause the other views to show the same part of the dialogue.

The Time Scale, cf. Figure 8.6, shows the synchronizations (as vertical lines) in linear time and the sound waveform, which is very useful when aligning coding points and media. The sound data may be adjusted with the sliders to the right and zoomed with the ZOOMIN/ZOOMOUT buttons. If you click on the ruler near synchronization it will become active (the active time and coding point will be shown in the text) and the following keys may be used:

P, N	Make the previous or next synchronization active
S	Synchronize the views to the active synchronization
? , ?	Change the time point value with 1/100:s of seconds
SHIFT+? , ?	Change the time point value with 1/10:s of seconds
SHIFT+CTRL+? , ?	Change the time point value with 1 second
DELETE	Delete the active synchronization
E	Edit the time point and/or coding point in a dialogue box

With some Java implementations these keys only work if the user clicks in the text field first.

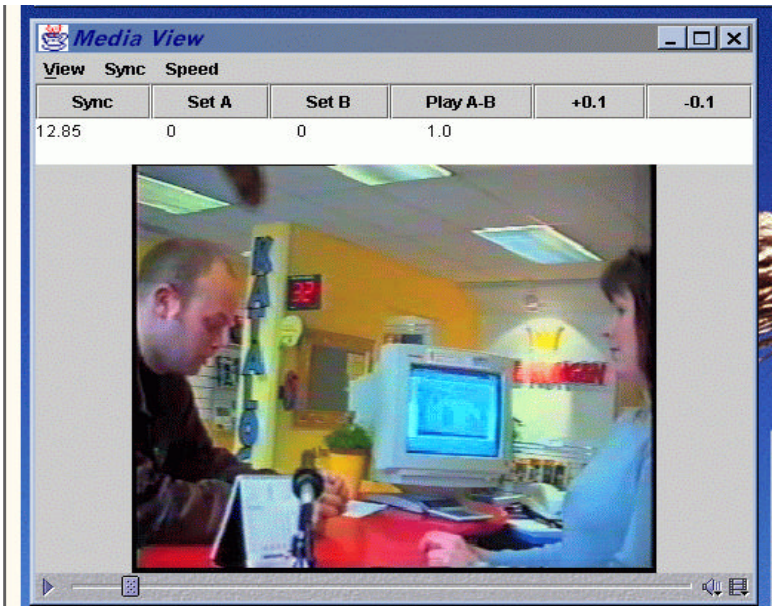


Figure 8.4. The Media View.

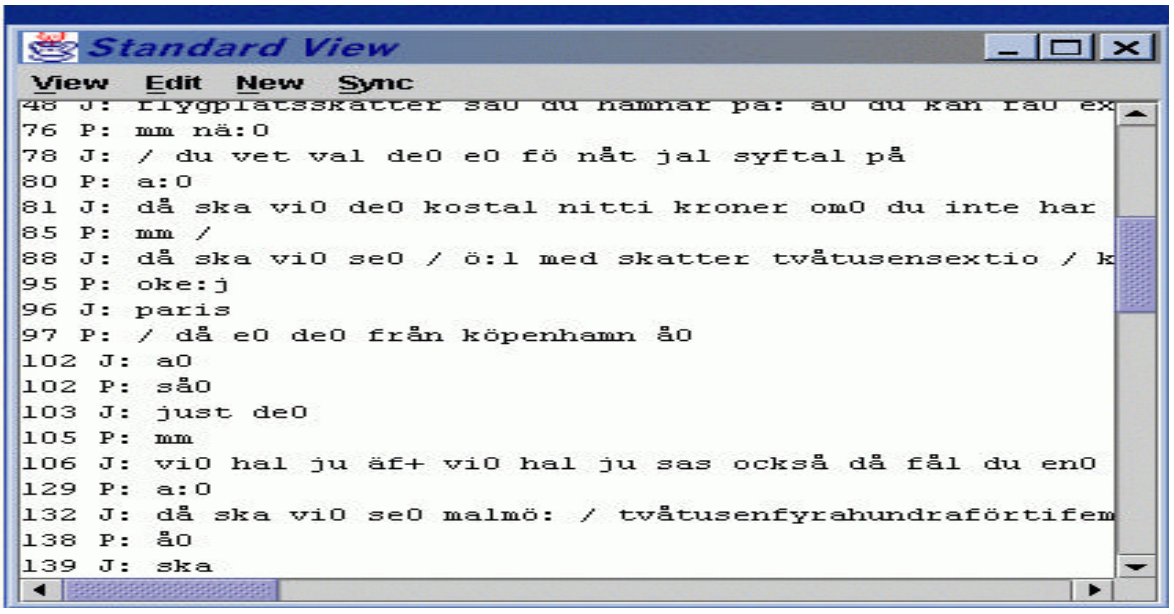


Figure 8.5. The Standard View of the Media View.

One important detail is that the views in Figures 8.4-8.6 can be synchronized to show the same sequence when the user scrolls in one of them. The internal state contains all information, so it is possible to have many views of the same kind, showing different parts of the dialogue. Changes made in one view will immediately cause a change in the internal state and the other views.

Other functionalities:

- One can import transcriptions, which fulfil the Gothenburg standard; (http://www.ling.gu.se/SLSA/dialog.html under the link “Transcription standard v6.2,1999”)
- One can synchronize begin/end points in utterances or codings, which means that if one chooses a piece of the transcription one can by pressing the synchronize button find the equivalent sound and/or video.

- One can search the media file (sound and/or video) in order to find the piece of video or sound, which equals the piece of dialogue one has found. The search function only works if the dialogue has been synchronised.

It is difficult to say whether the tool offers the functionalities that it promises, since it has not been tested hands-on, but according to the developers the tool is very unstable at the moment. At the moment one cannot annotate speech at multiple levels in the program, but if the project gets more funding the people behind Multitool plan to make this possible.

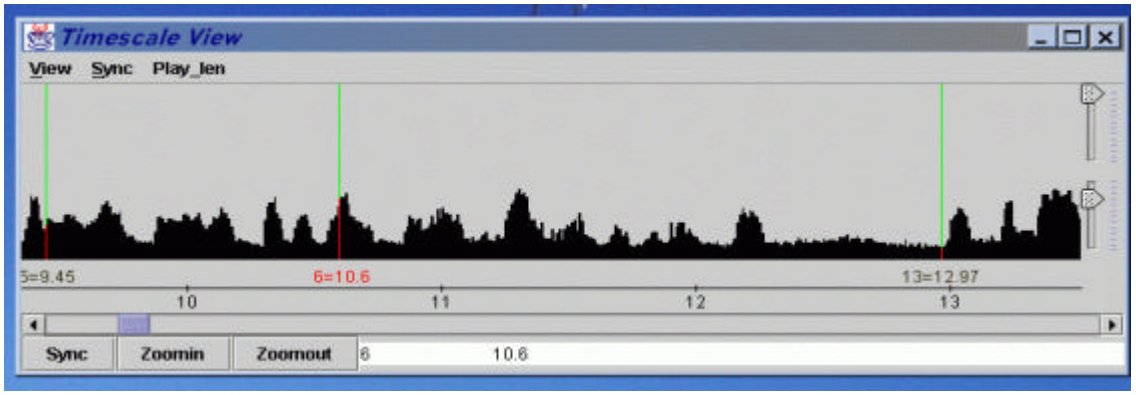


Figure 8.6. The Timescale View.

8.4 Conclusion

The idea behind Multitool, which was to build a platform in support of the establishment and use of multimodal spoken language corpora is in itself a very good idea. But due to the fact that the current version of Multitool is only partially implemented, it is not possible to conclude whether the Multitool software may become interesting as a tool for annotating multimodal corpora. The working part is the synchronisation between media files and transcription/coding. It is also possible to browse the data, and view the video/audio corresponding to different parts of the transcription. However, the coding and transcription have to be done in other programs. The platform will have to go through several tests and improvements before it can be expected to work properly. Whether this will happen depends on future funding.

8.5 References

Linguistic Annotation: http://www.talkbank.org/annotation.html

The Swedish Spoken Language Corpus at Gothenburg University - Tools: http://www.ling.gu.se/SLSA/tools.html

Abstract about Multitool: http://www.ling.gu.se/multitool/abstract.html

Users Manual: http://www.ling.gu.se/~leifg/doc/MultiToolUsersManual.pdf

Specification: http://www.ling.gu.se/multitool/specifikation.html

Screenshots and downloads etc.: http://www.ling.gu.se/multitool

The Gothenburg's standard: (http://www.ling.gu.se/SLSA/dialog.html under the link "Transcription standard v6.2,1999"

9 The Observer

9.1 Introduction

The Observer is a professional system for the collection, analysis, presentation and management of observational data. It can be used to record activities, postures, movements, positions, facial expressions, social interactions or any other aspect of human or animal behaviour as time series of tagged data. The tool has been designed in order to help people to record and annotate observational data, making it immediately available for quantitative analysis. The data can immediately be subjected to statistical analysis through recourse to a built-in data analysis module, in order to test an experimental hypothesis. The Observer comes under three standard system configurations: The Observer Basic is for data collection with a stand-alone computer; The Observer Mobile, for data collection with a handheld computer, and The Observer Video-Pro, for data collection from video.

The version reviewed here is the Observer Basic for Windows, version 3.0. A demo version can be freely downloaded from the Noldus web site (go to http://www.noldus.com, then to Products - The Observer - Short tour). The review here is based both on the description available from the website and from short hands-on experience with the demo. However, since the demo does not include the full range of functionalities offered by the Observer software, we have also added information on the full version of the Observer Video-Pro which is a more advanced tool but with which we have had no hands-on experience. The information on the Observer Video-Pro is drawn from (Noldus et al. 2000) and from the the Noldus web site (http://www.noldus.com). The Observer Video-Pro includes and extends the Observer software to an integrated system for collection, management and analysis of time-structured data from live observations, analogue or digital video tapes, and digital media files.

9.2 Software Design

The Observer has a modular structure where different program tasks have been distributed over multiple subprograms or modules. The Project Manager is the center of all modules, and it controls creation and management of projects, data import and export, and activation of other program modules, including the Configuration Designer, the Event Recorder and the Analysis Procedures.

The *Configuration Designer* module allows the user to specify what he/she wants to observe and how this is to be recorded. The *Event Recorder* module is the environment where the actual observations are performed. The data is automatically saved in observational data files. In the *Data Analysis* modules the collected data can be analysed in various ways.

The current versions of The Observer uses a graphic interface and runs on Windows computers as well as compact hand-held computers for use in the field. Subject to availability of hardware resources, two or more modules of The Observer can be run simultaneously. This allows the user to display the results of separate analyses simultaneously on the screen.

The Observer Video-Pro is an integrated system, comprising various software and hardware components. The software consists of the *Base Package for Windows* and the *Support Package for Video Analysis*. The Base Package includes software functions to design research projects, create observational coding schemes, collect 'live' observational data, manage data files, and perform data analysis. The Support Package adds to this the functionality to collect, review, edit and summarize observational data from analogue and digital video tapes and digital media files. The software has been written in 32-bit object-oriented C++, using Microsoft Visual Studio 6.0 with MFC and DirectX libraries. The code has been optimized for Windows 95 (and up) and NT 4.0 (and up).

9.3 Platform requirements

9.3.1 The Observer Basic

- Processor (CPU): Pentium90 or higher.
- Internal memory (RAM): 16 MB minimum, 32 MB recommended.

- Hard disk: at least 100 MB free space.
- Screen resolution: 640x480 minimum
- Operating system: Windows 95, Windows 98, Windows NT 4.0, Windows 2000.
- Network operating systems: Novell NetWare, Windows NT Server.
- Printerport or USB port for hardware key.
- CD-Rom drive

9.3.2 The Observer Video-Pro

- Processor (CPU): Pentium90 or Higher.
- Internal memory (RAM): 16 MB minimum, 32 MB recommended.
- Hard disk: at least 100 MB free space.
- At least one free 16-bit ISA slot or PCI slot (for time code reader).
- Other boards (video overlay board, etc.) will need additional ISA or PCI slots.
- Free serial port (RS-232 port) when using VCR.
- Screen resolution: 640x480 minimum, 800x600 or higher recommended.
- Monitor: with 1024x768 or higher screen resolution we strongly advise the use of a 17" or larger monitor.
- Operating system: Microsoft Windows 95, Windows 98, Windows NT4.0 Windows 2000.
- Printerport or USB port for hardware key.
- CD-ROM drive.

In addition, if one uses digital media files on hard disk, CD or DVD:

- Processor (CPU): Pentium 133 minimum, Pentium-II or higher recommended.
- Internal memory (RAM): 32 MB minimum, 64 MB recommended.
- Hard disk: sufficient free space for temporary storage of digital media files (with MPEG-1: approx. 600 MB/hr, with MPEG-2 approx 2,4 GB/hr).
- At least one free 32-bit PCI slot for the video capture board.
- Other boards (sound card, video overlay board, MPEG-2 decoder etc.) will need additional ISA or PCI slots.
- CD writer or DVD-writer, to copy digital media files from hard disk to CD or DVD.

9.4 User interface

The Observer starts off with the **Project Manager window**, designed so as to allow the user to select one specific annotation project, if existing, or to create a brand new one (Figure 9.1). An annotation project is an operational equivalent of the MATE notion of a project, cf. Figure 6.2. It consists of a set of files, containing either configuration information, annotated observational data or analysis data which all relate to the same studycf. Figure 9.2.

The **configuration file** contains all information about the methods and variables to be scored of a particular setup. This file is used during event recording and for the analysis of data files.

Here is a concise list of the various aspects of a configuration:

- Data Collection Method - Sampling method and the number of actors.
- Timing Method - Settings that affect the timing of events and observations.
- Independent Variables - These are variables of which the value does not change during the course of an observation.
- Dependent Variables - The outcome measures which are the behaviours that you score. The Observer defines behaviour by the following items:
    - Subjects;
    - Behavioural Elements;

- Modifiers associated with behavioural elements;
- Channels - Combinations of a focal subject and a class of behavioural elements.

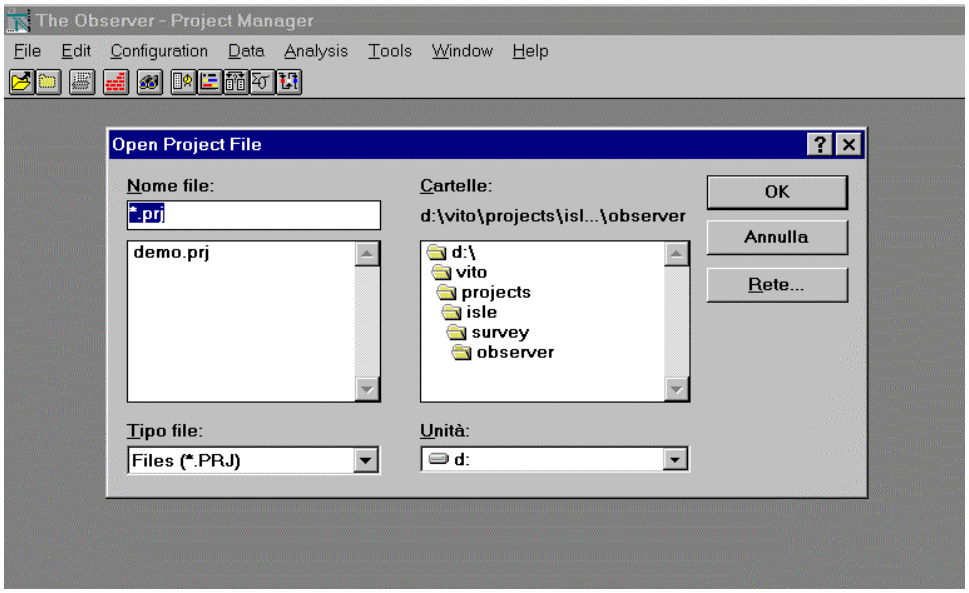


Figure 9.1. The Project Manager Window in the Observer Basic version 3.0.

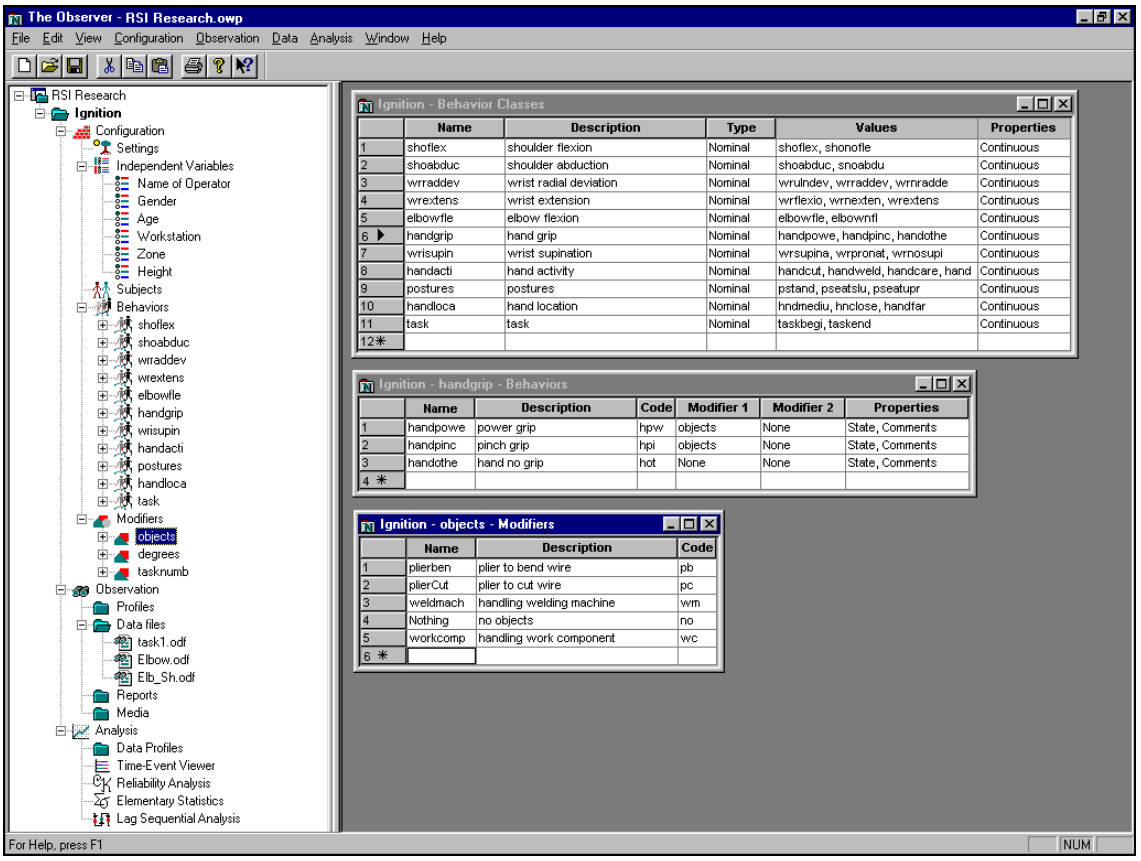


Figure 9.2. Screen of The Observer Basic version 4.0 showing the structure of a project. The pane on the left shows the workspace “RSI research” as the highest level of organization. The workspace contains one project named “Ignition”. The project is divided in Configuration, Observation and Analysis, which represent the three stages in an observational study. The windows on the right show details of three items of the configuration: the 11 classes of behavioural elements, the elements in the class “handgrip” and the class of modifiers named “objects”.

A configuration file is created interactively through use of a separate **configuration window** which gets activated by clicking the configuration button in the Project Manager window. Although the window offers special distinct functions to set up the relevant parameters, novel users can find this way of dealing with a configuration file not immediately intuitive, at least at first blush. Parameters can be defined in any order, but as some settings logically presuppose some others, such a leeway can lead a novice to confusion.

Collecting observational data with The Observer is done with a separate module called the **Event Recorder**, which has a window of its own (Figure 9.3). It is assumed that the user has already created a configuration file, and integrated this file in a project. The Event recorder window can be customised by the user in a variety of ways. It is possible to record the same time series of events at different independent levels of annotation, which are kept aligned through reference to the same time line. This is made possible by defining multiple **channels** of recording, each of which is annotated independently. Annotated records can then be edited, analysed and displayed simultaneously, by projecting them onto the same time axis.

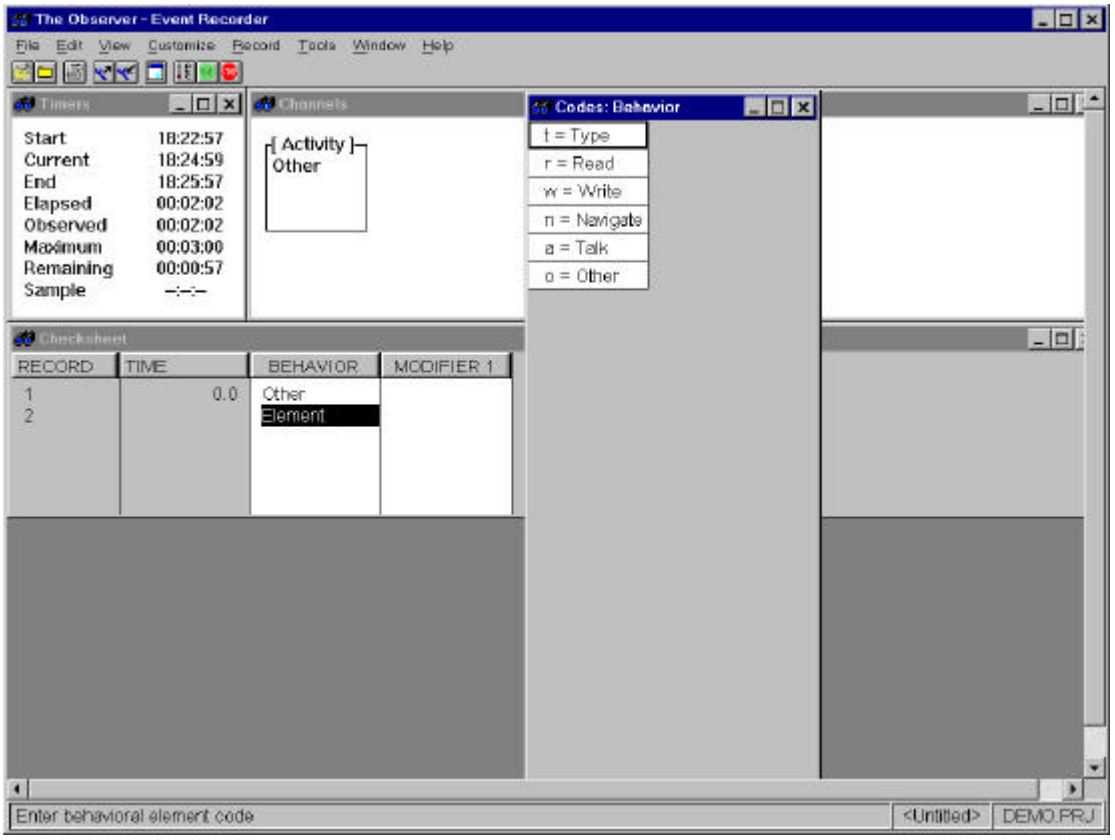


Figure 9.3. Event Recorder in the Observer Basic version 3.0.

The demo shows no operational example of multi-channel data recording, so it is impossible for us to judge how intuitive and user-friendly the process is.

Annotated data can be edited to amend errors or damages before data analysis is started. This can be done off-line within the Project Manager window, and on-line, during annotation, in the Event Recorder window.

Event recording with the Observer Video-Pro can be done in three modes. In a 'live' setting, the internal computer clock, accurate to 1/18 second, is used as the event timer. With data collection from (analog or digital) tape, a time code in each video frame allows event timing at 1/30 (NTSC) or 1/25 (PAL/SECAM) second resolution. Certain time code generators and readers allow single-field precision, i.e. 1/60 or 1/50 second, respectively. The recommended time code generators produce vertical-interval time code (VITC), which can be read at different playback speeds and which leaves the two audio tracks available for other use (e.g. verbal comments). With data collection from digital

media files, frame numbers in the file are used as the time basis. During operation, The Observer translates these numbers to hours, minutes, seconds and fractions of seconds.

9.5 Functionality

The main functionalities of The Observer concern data collection, data analysis and export functions.

9.5.1 Data collection

For data collection, it is possible to enter data directly into a PC or handheld computer, or code events from video tape. Prior to actually starting an observation, the user has to enter a name for the data file. One can also select an existing data file with its associated video file to insert records into the data file, thereby making the data file increasingly detailed. Next, the user assigns a value to each of the independent variables predefined in the research design, positions the media file at the point where data collection should commence, and finally starts the observation. During data acquisition actors, behaviors and modifiers are scored by typing predefined key codes or clicking codes on the screen. At each valid key press the program calculates the elapsed time from the start of the observation and logs the time-stamped event record in the observational data file (Figure 9.4). If the user enters an undefined code, an alert message is displayed. One can also use a mouse or a computer pen as the input device. During observation, the user can add notes and comments, which are stored together with the data. The look-and-feel of the program can be customized to the user's own taste, thus turning The Observer into his/her personal research partner.

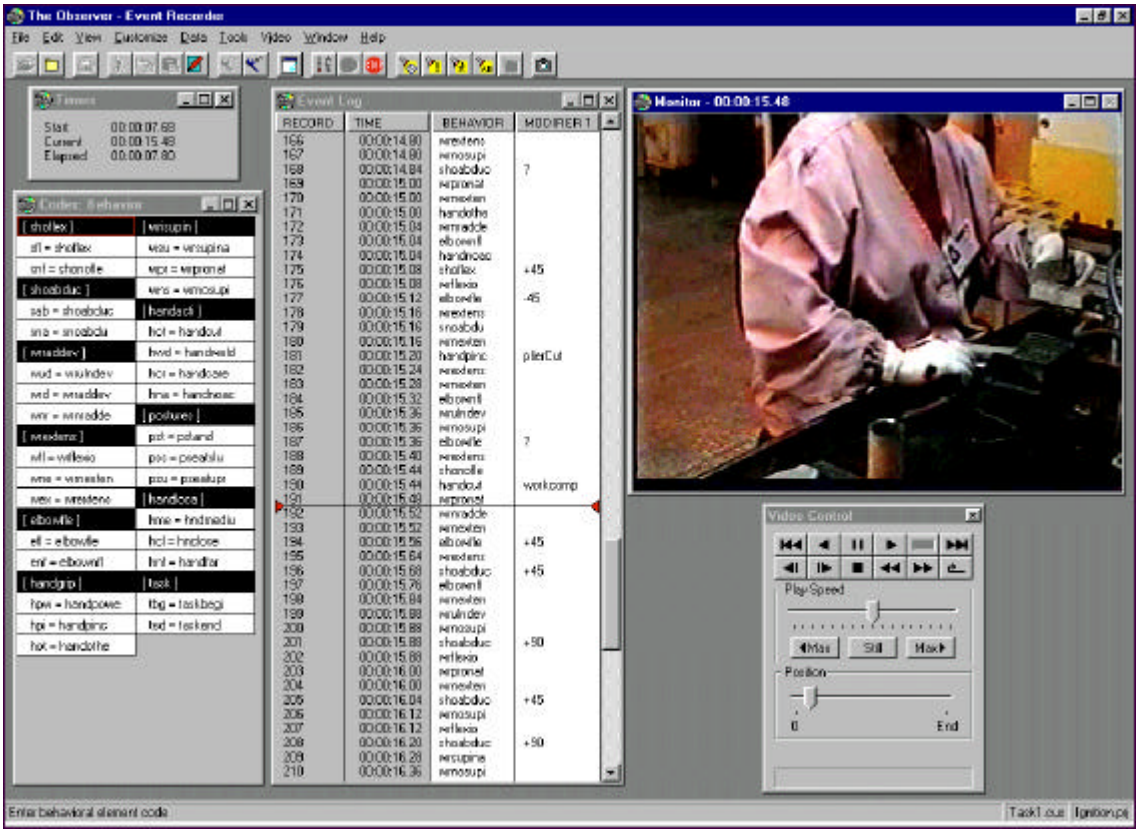


Figure 9.4. The main application window during an observation session. The Event Log window contains the chronological list of event strings, each with a record number, time stamp, behavior and modifier. The horizontal cursor indicates the time that corresponds with the current position in the video file. The Monitor window displays the digital video file, with the current time in its title bar. The Codes window shows all predefined behavioural elements with their key codes, grouped in classes. The Video Control toolbox emulates the front panel of a VCR and allows the user to control the video display. The Timers window displays the start time of the observation, the current time of the video file and the elapsed time of the observation. This screen is just one possible screen layout; presence, size and position of each window are fully customizable. After ending the observation, the data file is saved, added to the current project, and available for data analysis.

9.5.1.1 Special functions available during an observation session

To facilitate data entry, retrieval and summarization, a number of special program functions have been developed. A small selection is discussed below.

Free-format annotation. In addition to scoring predefined events, one can enter three types of free-format annotations in the data file. First, one can insert marker records with short notes attached. Secondly, for more extensive time-stamped notes, The Observer offers a special Notepad window. Thirdly, one can enter comments for each record, as an extra column in the data set.

Video control. The playback of the video tape or file can be controlled via menu options, shortcut keys or on-screen buttons. Play forward, play reverse, rewind, fast forward, pause, stop, frame-by-frame jog and various shuttle speeds forward and reverse are supported. Which options are actually enabled depends on the video source used. The Observer Video-Pro comes with built-in drivers for a wide range of VCRs and media file players. One can jump to any position in the digital video file using a position slider. A Quick Review option allows one to jump back a user-defined interval followed by slow-motion playback at user-defined speed, triggered by a single mouse click or key press.

Finding events and video images. The user can search for a specific time stamp, event, marker, or text string in the data file (Figure 9.5). Provided that the computer controls the video source (tape or file), the corresponding video frame will be displayed as well. One can also define a 'roll-on interval', to see the events that led up to the event that was searched for. While a search operation on tape may last seconds to minutes, access to a particular event and corresponding video frame in a digital media file is virtually instantaneous.

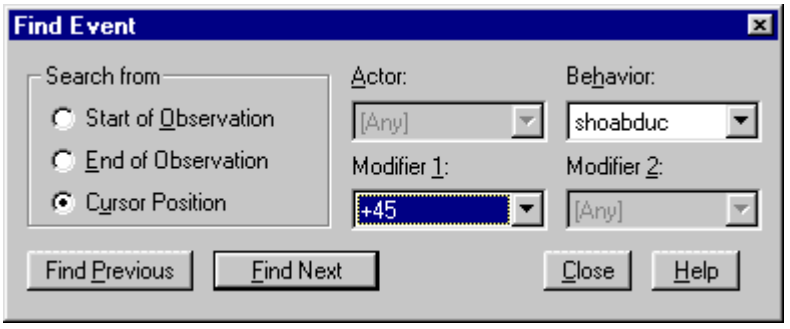


Figure 9.5. The Find Event dialog box. The Actor, Behavior, Modifier 1 and Modifier 2 drop-down list boxes hold all predefined actors, behaviors and modifiers. Finding a specific event will also display the corresponding video image.

Video play list. In a typical observational study, comprising many hours of video recording, one rarely replays all video footage when presenting one's findings. One rather shows a few short clips to illustrate the most significant results. Creating a video compilation manually can be a very time-consuming operation, which is why only few people make use of this technique as part of scientific presentations. One can use special video editing systems for this purpose, but these are often complex to use and offer a lot of functionality beyond what is needed by the behavioral researcher. To resolve this, The Observer Video-Pro allows the user to create a *video play list*, i.e. a series of video episodes placed in a user-defined order for playback. The user creates this list by simply dragging records from the event log to the video play list, after which sequence, duration and other properties of the clips can be modified. The episodes can be played on screen (or sent to a computer projector), or copied to a compilation video tape or CD. Single video images can also be captured and saved as disk files, for use as illustrations in documents, slides for presentations, etc. With the Video Highlights Module it is possible to create a Video Play List (VPL) automatically. The user can search for actors, behaviors and modifiers, and each time a particular combination is found in the Event Log, the associated video episode can be added to the (VPL).

Event summary. For applications where an accurate numerical analysis is not required, The Observer offers an *event summary*. This is a table with a user-defined selection of events from one or more data files. Events can be edited, comments can be added and records can be ordered using various sort

keys. This tool is especially useful for qualitative and exploratory analysis of selections of one or more data files. The results can be printed with a header and footer, or saved in a file.

9.5.2 Data analysis

Before subjecting observational data to a specific analysis, one can filter them at several hierarchical levels. The following filters are available:

- *Observations*. One can select one or more observational data files from a project. Files can be selected manually or based on values of independent variables (e.g. all files for which *sex=male*).
- *Time window.* This refers to the section of the data file to be included in the analysis. Start and end criterion can be either time-based or event-based, e.g. from *time=30* to *event=John looks.* One can also run an iterative analysis across multiple time intervals (e.g. 1-minute blocks in a 30-minute experiment).
- *Nesting levels*. These are used to measure the co-occurrence of events and states in different classes. One can combine nesting conditions with boolean operators to form complex queries and answer questions like "How often did the operator reach for a tool with his right hand while holding an object in his left hand?".
- *Events*. One can select the actors, behaviors and modifiers for which statistics must be calculated. Each element in the data record (actor-behavior-modifier1-modifier2) can be replaced by a wildcard, resulting in 2^4=64 different selection filters.

Several analysis options are available. The collected data can be explored in time-event tables and plots, or reports can be generated with statistics on frequencies and durations, the sequential structure of the process or the co-occurrence of events. Results can then be displayed on screen, printed on paper or saved in a file on disk.

- *Time-Event Table:* A time-event table is a chronological listing of all recorded events, sorted in columns for the various subjects and classes of behaviour. It visualizes the change of states and the occurrences of events in the course of time.
- *Time-Event Plot:* This is a graph in which observational data are plotted against time. Time-event plots give you a quick overview of what happened and when. You can define the content of each trace and zoom in on temporal details. You can select the drawing style, and also let the timeline scroll on the screen. You can even let The Observer emulate a traditional strip-chart recorder.
- *Elementary Statistics:* This provides a wealth of descriptive statistics on the frequency and duration of events or states. Statistics can be calculated for subjects, individual behaviours, modifiers, and combinations of those. You can split or lump categories, using nesting levels to create complex queries, define (iterative) time windows, average data across observations, and much more!
- *Lag Sequential Analysis:* This technique examines how often certain events are preceded or followed by other events. For instance, in a usability evaluation, The Observer tells you how often a certain operation is followed by an error. You can choose between state lags or time lags and set the lag length.
- *Reliability Analysis:* Reliability analysis measures the level of agreement between pairs of raters. Reliability is expressed as an index based on the degree of agreement between the scores in two data files. Whether an entry in an observational data file is scored as a match or as an error is modified by the user. You can set the tolerance level by defining a time window within which entries are scored as matches.

9.5.3 Export functions

For additional calculations and inferential analysis (hypothesis testing), the user can export the summary tables to most spreadsheets, databases or statistics packages available on the market. The Observer formats the output for the package of your choice.

9.6 Conclusion

The modular software architecture maximizes flexibility in data collection and analysis. The tool adds value to video material by anchoring observational data to video frames, by validating entered data against coding schemes, and by tightly integrating data collection with numerical and graphical data analysis.

In spite of the comparatively simple design of each window and the wisely sparse recourse to buttons, default window configurations are not very telling and a more consistent use of simple on-line help facilities would be beneficial. In particular, the event recorder window is made cumbersome by the simultaneous display of different channels, which, albeit operationally useful, can at first be confusing. Nonetheless, the tool is a most notable example of efficient, comparatively user-friendly and flexible working environment for recording observational data in a complex way. In particular the key-idea of "project" is considerably advanced, and the overall annotation structure, articulated over multiple channels, makes the software design a natural candidate for extension to multi-modal annotation.

9.7 References

http://www.noldus.com/products/observer/index.html.

Noldus, L.P.J.J., Trienes, R.J.H., Hendriksen, A.H.M., Jansen, H. And Jansen, R.G.: The Observer Video-Pro: New Software for the Collection, Management and Presentation of Time-structured Data from Video Tapes and Digital Media Files. Behavior Research Methods, Instruments & Computers, vol. 32, 2000, 197-206.

10 SignStream[1]

10.1 Introduction

SignStream is a multimedia database tool for analysis of linguistic data captured on video, developed as part of the American Sign Language Linguistic Research Project. The tool was originally intended to be applied in research on Sign Languages, where video is the primary medium for collecting data, to overcome the limitations of conventional representations of signed language data in written gloss form. In spite of its originally being oriented towards representation of signed language data, in principle SignStream can be used for any kind of linguistic research that relies on video data, including language acquisition, bilingualism, neurolinguistics and language disorders, phonetics and phonology, sociolinguistics, discourse and narrative studies, pragmatics, analysis of the gestural component of spoken languages; and computational linguistics.

SignStream provides a unified computing environment for manipulating media files and for transcribing and analyzing utterances. The program allows simultaneous access to digital video and audio files along with representations of the data in digital video and linking specific frame sequences to simultaneously occurring linguistic events encoded in a fine-grained multi-level transcription. Items from different fields are visually aligned on the screen to reflect their temporal relations. SignStream thus greatly simplifies the transcription process and increases the accuracy of transcriptions (by virtue of the link of linguistic events with video frames), also enhancing the researcher's ability to perform linguistic analyses of various kinds.

The review is mainly based on the description of SignStream Version 2.0 available at the SignStream web site. A demo version of the tool can be downloaded and evaluated for a 30 days period. The software is distributed on a non-profit basis to students, educators, and researchers for non-commercial use only.

10.2 Software design

No data is available.

10.3 Platform requirements

SignStream version 2.0 runs on MacOS computers running system 8.0 or later and QuickTime 3.0.2 or later. Other requirements:

- a minimum of 3MB of RAM plus some additional free memory;
- video data must be available in digital format.

10.4 Interface

SignStream provides an intuitive interface for viewing, manipulating, and coding data, including visual representation of simultaneity of overlapping events. A SignStream *utterance* associates a segment of digital video with a transcription. The video and transcription are displayed in a video window and a gloss window, respectively, as illustrated below. These are the two main environments of the user interface. An example of the two environments is shown in Figure 10.1.

10.4.1 The video window

The video for an utterance is displayed and manipulated in the video window. The video is a standard QuickTime movie and thus can be manipulated through the standard set of controls provided with QuickTime, among which, for instance, frame-by-frame advance and rewind functionalities, and a scroll bar to aid in positioning the video at any point in time. SignStream also provides a variable

[1] This material is adapted from documentation available at the SignStream web site (see http://www.bu.edu/asllrp/SignStream). Thanks to Carol Neidle for comments on earlier drafts of this document.

speed controller, which allows the user to change the speed at which the video is played. The video controls appear either in the video window or in the gloss window, depending on which one is active. Version 2.0 also provides capability for associating multiple (up to 4) synchronized video files as well as one synchronized AIFF sound file.

SignStream provides specialized capabilities for manipulating the video in relation to the linguistic units defined within any data field. For example, the user can select an item within the gloss window (see below) and then perform any of the following actions: establish its beginning and end frames, position the video at its beginning or end frame, or play the video clip associated with the item.

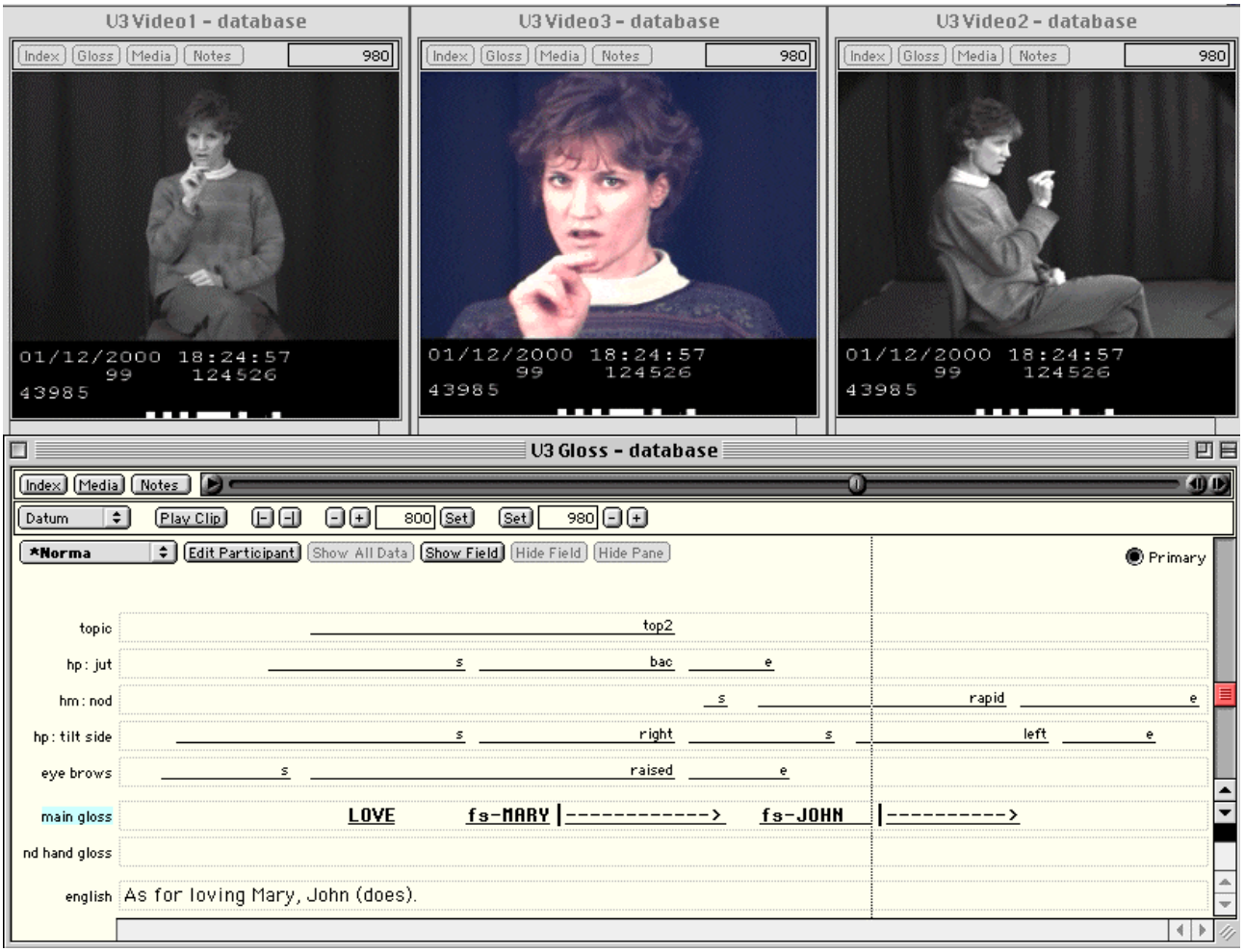


Figure 10.1. SignStream Multiple Video Window.

10.4.2 The gloss window

The gloss window provides the core functionality for manipulating transcriptions of utterances in the database. A number of fields is available to encode different kinds of linguistic or kinesic events, such as, for instance, hand, eye and eyebrows movements. Items in different fields are linked to specific frame sequences of the video, and are aligned on the screen to reflect their temporal relations. The SignStream program provides a set of predefined fields specifically designed for representing sign language data, including: the main gloss, the non-dominant hand gloss, non-manuals (such as eye gaze, eye brows, head movement, topic, yes/no question), part-of-speech tags, and English translation. However, the program also allows the user to define new fields and new field values.

The gloss window's display can be tailored in a number of ways. For example, the user can select which fields should be displayed, rearrange the fields' relative positions on the screen, and change the colour that is used to display the various fields.

The gloss window is also the main input environment, where annotation of video frames can be entered. This happens in a number of ways: by typing, by drawing lines, by selecting values from menus, etc. For each linguistic unit that is entered, the user establishes the start and end frames by 1) selecting that linguistic unit 2) locating the relevant frames of the video and 3) clicking on the appropriate "Set" button. Data items can be edited or removed at any time.

Non-manual information is entered by the user by drawing a line within the appropriate non-manual field. The user clicks at the starting point of the non-manual and drags the endpoint of the line to the desired position. As the user extends the line, the video is automatically updated to show the frame corresponding to the current endpoint. When the user releases the mouse button, the line is completed at that point, and a menu appears from which the user can select the value to be associated with the line that was just drawn. The user can later edit the start and end positions for the non-manual item, either by dragging an endpoint of the line, or by using the same Set buttons that are used for the main gloss and other fields.

10.4.3 The Media Controller Window

A media controller window (see Figure 10.2) allows for selection and control of media files. To select a movie or sound file, the user must simply click on the "Select" button in the media controller window and then locates the desired file.

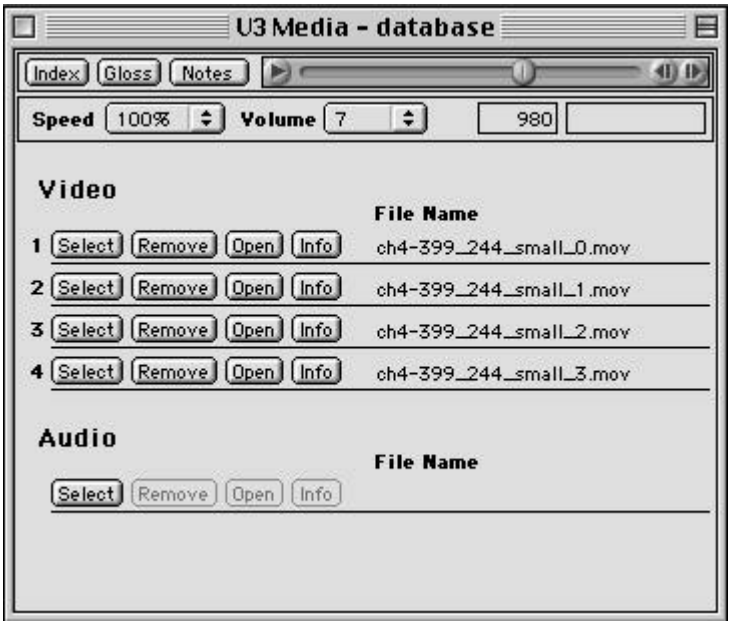


Figure 10.2. SignStream Media Controller Window.

10.4.4 The Audio window

For sound files, a window displaying the sound wave is available, as Figure 10.3 illustrates. The media alignment indicator (shown by the green vertical line) indicates the current position of the current frame in each of the media windows (i.e., the video and audio display windows). The user can play the media at varying speeds and make use of normal QuickTime controls.

10.5 Functionality

The main functionalities of SignStream include:

- browsing of video data;
- annotation of video data according to simultaneous layers of description;
- automatic alignment of simultaneous linguistic events: an important feature of SignStream is the automatic spatial alignment on the screen of items from different fields that co-occur

temporally. This feature allows the alignment of information which is co-extensive in time but have different individual durations.

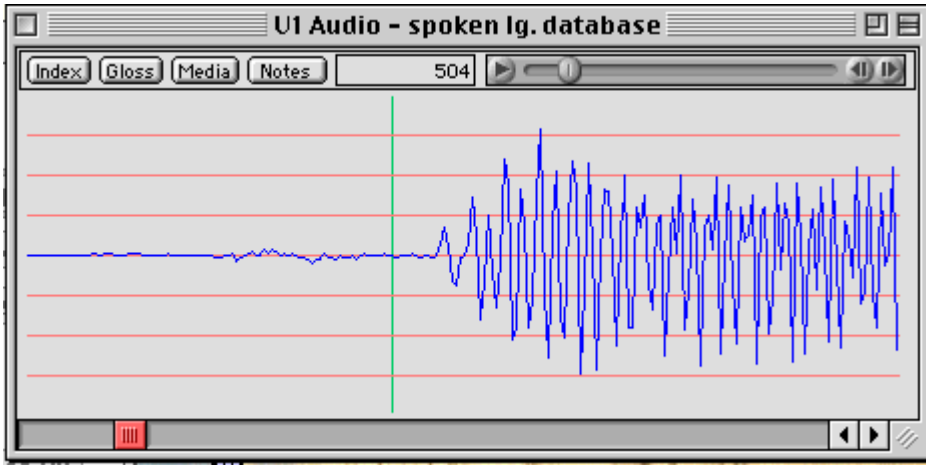


Figure 10.3. SignStream Audio Window.

A SignStream database is composed of utterances, where each utterance is associated with a segment of video together with a detailed, multi-level transcription of the language data contained in the video. Simultaneous linguistic events are coded in separate fields, and are connected to particular segments of the video. The program aligns the fields so that simultaneous events in different fields are aligned vertically on the screen. Multiple utterances can be viewed at the same time: for example, it is possible to display two utterances side by side, in order to compare their associated video segments and/or their transcriptions.

Other relevant functionalities are:

- representation of both start and end points of coded items;
- multiple movies can be incorporated into a single database and potentially overlapping segments of video can be coded as distinct utterances;
- multiple discourse participants may be coded in separate viewing areas in the gloss window, called "participant panes", cf. Figure 10.4 (this figure refers to the older version of the software). One or more participant panes may be open at the same time, and all data in all panes are aligned to each other and linked to the video. Each participant is associated with a profile, containing relevant background information.
- association of multiple synchronized video files;
- search functionalities: SignStream allows the user to carry out sophisticated search of SignStream databases. The user first builds a search query, by combining search operators with data specifications. Queries can be constructed either by typing or by selecting from menus. In addition to standard Boolean combinatorial operations, SignStream provides a number of operators that are relevant to the particular type of data represented in the databases. For example, the WITH operator can be used to search for combinations of data that are co-extensive, permitting a user to search for utterances containing, e.g., an index sign co-occurring with eye gaze. Searches can be conducted in a number of sequential stages, where subsequent stages search over the results found in a previous stage, cf. Figure 10.5. This allows the user to easily refine a search to narrow in on the data of interest. Searches (both queries and results) may be saved to a file for later use.

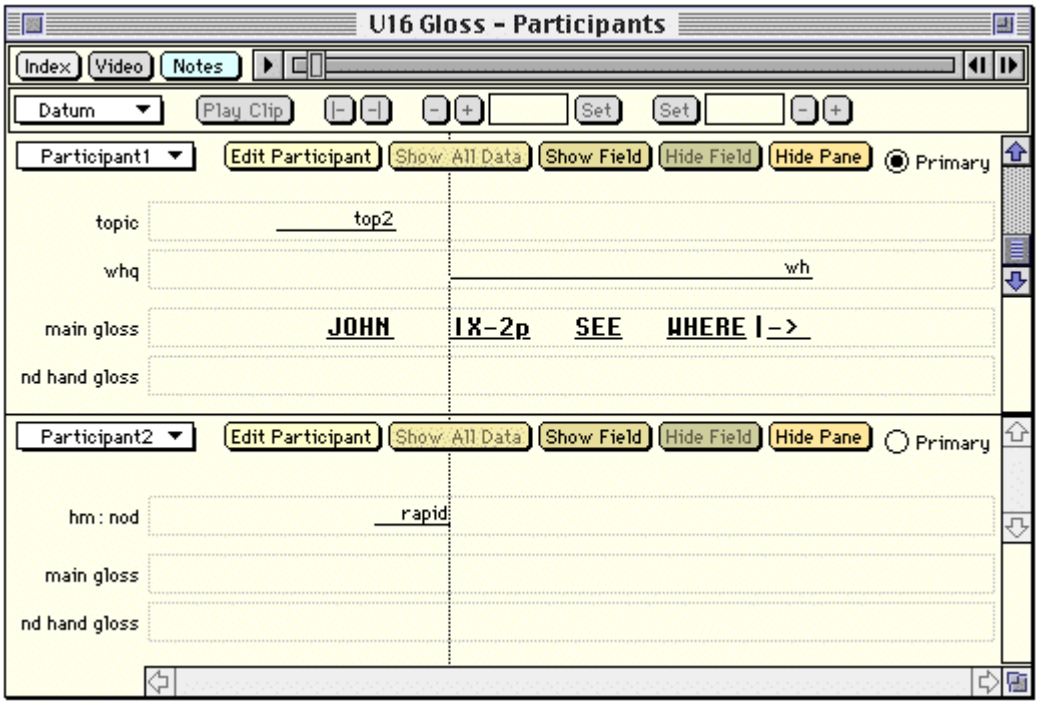


Figure 10.4. SignStream Participant Panes.

10.6 Conclusion

SignStream appears to be a user-friendly tool for effective use in sign language data annotation. Many of the functionalities provided are well suited for use in annotation of natural interactivity and multimodal data.

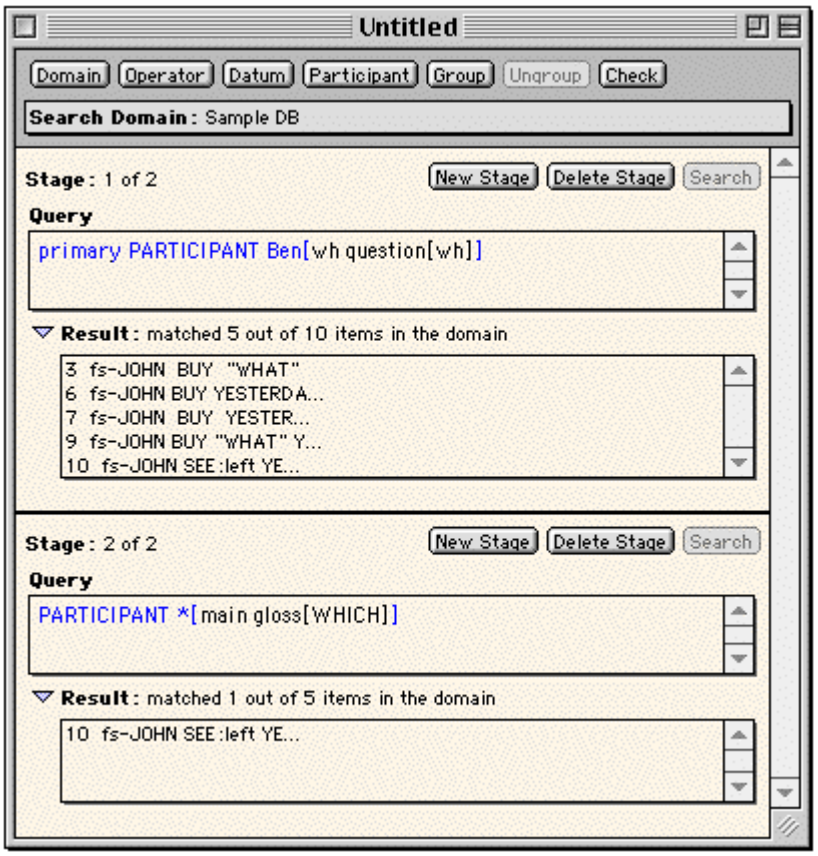


Figure 10.5. SignStream Search Window.

10.7 References

The SignStream web site: http://web.bu.edu/asllrp/SignStream/

MacLaughlin, D., C. Neidle, and D. Greenfield. 2000. SignStream™ User's Guide, Version 2.0. American Sign Language Linguistic Research Project, Report No. 9, Boston University, Boston, MA. http://www.bu.edu/asllrp/reports.html#RPT9.

Neidle, C. 2000. SignStream(TM): A Database Tool for Research on Visual-Gestural Language. American Sign Language Linguistic Research Project Report No. 10, Boston University, Boston, MA.

11 SmartKom

11.1 Introduction

This contribution provides an overview on the general system, the data collection efforts and its requirements, and the current state of markup in SmartKom. Since this is an ongoing effort, many details are still not defined. Therefore, this contribution describes the status quo which most probably will change in the future.

The data contained in this report rest mainly on personal communication and is partly drawn from the three internal reports listed in Section 11.8.

The responsibility for the contents of this study, however, lies solely with the author.

11.2 General Overview of the SmartKom-Systems

SmartKom's goal is research in the area of multi-modal interactions, leading to the development of a self-explaining, user adaptive interface to various applications. The project is guided by the aim to merge the advantages of dialog-based communication with the advantages of a mixture of graphical user interfaces and gestic and mimic interaction. The project - a joint initiative involving information-technology companies, universities, and research organisations - is funded by the German Federal Ministry for Education and Research (BMBF) with additional funds by the industrial partners. The consortium consists of DFKI GmbH, EML GmbH, Universität Erlangen-Nürnberg, ICSI, Universität Stuttgart, Universität München, MediaInterface GmbH, Philips GmbH, Siemens AG, and Sony GmbH.

SmartKom develops core functionalities for intelligent communication assistants which analyse language, gesture and miming. The assistants interpret these input modalities in the context of the dialog and initiate actions accordingly.

The abilities of SmartKom will be tested within three real application scenarios:

- SmartKom-Public can be looked at as an extension of traditional public telephone booths. The user can enter the booth, identify herself using biometrics – is she wants do use personalized services –, and use information services like WEB, reservation agencies, calendar etc.
- SmartKom-Mobil is a permanent personal companion which enables the user to access information everywhere at any time. The system is mobile, but can be connected, e.g., with the car's information infrastructure. The user has information access anywhere she wants.
- SmartKom-Home/Office - a personal computer work station - will continually develop into the command center of an integrated living environment. In contrast to the two other scenarios, the system employs not only a „lean-forward" mode, where the user concentrates on the interaction with the system, but also a „lean-backward" mode, where the system is somewhere in the user's living or working room. It is activated from time to time, e.g. to operate some consumer electronic device, without paying close attention to graphical output of the system.

Thus, SmartKom examines the flexible integration of functionalities and hardware into a homogeneous system. The interaction will be uniform, comfortable and intuitive, mainly based on natural dialog. The SmartKom architecture is shown in Figure 11.1.

More information - mainly in German - can be found at http://www.smartkom.org/. There you can also find first realization sketches for scenario adapted systems.

11.3 Modalities in SmartKom

Interaction in SmartKom exploits multiple modalities:

- speech input and output
- deictic, non-manipulative gestures on visualized information content
- mimics

- biometrics

One basic idea of SmartKom is to combine modalities to ease the access to complex data sources by combining mainly two modalities, speech and gesture. SmartKom does not follow the WIMP (Windows, Icons, Mouse Pointer) metaphor in its use of gestural interaction. Rather, the (deictic) semantics of a gesture is to be recognized, and processed, either combined with speech input or in its own right. To enable rich gestural interaction the presentation must not rest only on the direct manipulation metaphor. The presentation is adapted to the semantic nature of interaction.

While these two modalities allow complex interactions, the last two modalities are used to either identify the user using biometrics, where signatures and hand shapes are under consideration, or to identify the mood of the user by classifying the mimics and head movements. A classification system tries to identify positive, neutral, or negative mood towards the ongoing communication. This feedback can e.g. be used to assess the acceptance of the ongoing dialog.

For speech I/O, SmartKom uses a variety of microphones to be seamlessly integrated in the various scenarios.

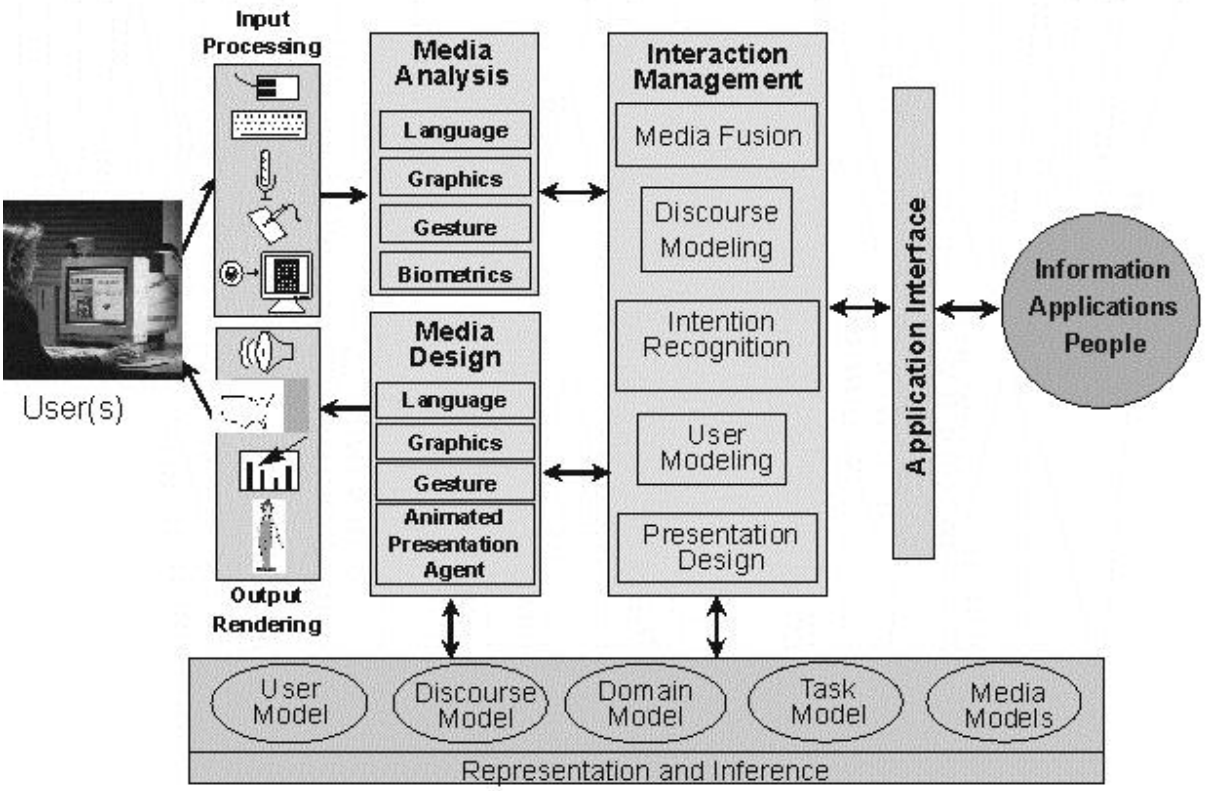


Figure 11.1. SmartKom architecture.

11.4 Data Collection

For training, test, and evaluation purposes the developers need data collected as realistically as possible. This data collection is performed at the Ludwigs-Maximilians Universität München.

Figure 11.2 shows the data collection prototype. The main unit consists of the SIVIT ^{TM} virtual touch-screen. On top is the unit which contains the data projector and the hand-movement recognition hardware. The visualized content is projected on the surface in front of the test person. The surface contains a graphic tablet which allows also direct interactions on the surface. This interaction is possible for SmartKom-Home and SmartKom-Mobile scenario WOZ dialogs.

Between the two poles of the supporting frame, you can see the mimics camera and an attached microphone. The four lights provide a constant brightness for the mimics camera. Additional microphones are at the further end of the projection panel and attached to the test person. There are a total of 9 microphones.

During the WOZ dialogs, all signals are recorded, including background sounds, two video streams from monitoring cameras, and a video stream for the visualized content. They are stored as Quicktime movies. Quicktime was chosen at the start of the project as standard due to the available software at that time. It allows for the time alignment of all streams in a widely accessible format.

11.5 Annotation

Currently, the annotation of the various modalities in SmartKom is separated in two tasks, namely the annotation of the audio signal and the annotation of mimics and gestures.

11.5.1 Transliteration and annotation of audio data

The annotation of the audio data rests on the convention and tools developed in the Verbmobil project. The words and paralingual sounds are transcribed and distributed either as single text files (so called `trl` format), or as partitur files (`.par`), where each level of transcription and annotation is written in a separate track, and time alignment information for each word is readily available. See http://www.phonetik.uni-muenchen.de/Bas/BasFormatseng.html for a further description of the formats.

Figure 11.2. The SmartKom system for Data Collection.

```
; CDR:
; TRV:
; Dialog: w001_pk
; Tonqualitaet: URB:normal; SMA:leise
; ATMO: Demovideo
;
w001_pkd_001_SMA:
<:<#Rascheln> her<Z>zlich:> [NA] will<Z>kommen [NA] bei <:<#> ~SmartKom:> [PA] [B3
fall] <P> wie<Z> [NA] kann ich Ihnen hel<Z>fen<Z>  [PA] [B3 fall] ?

w001_pkd_002_URB:
      ich w"urd' gerne ins Kino [PA] gehen [B3 fall] .

w001_pkd_003_SMA:
ich zeige [NA] Ihn<Z>en <!1 Ihn'> eine <:<#>"Ubersicht<Z>:> [NA] [B2] der<Z> [B9]
Programme<Z> [NA] [B2] in den ~Heidelberger [NA] Kinos [PA] [B3 fall] .
```

Figure 11.3. A trl example.

An example of a transliteration is shown in Figure 11.3. The header contains information, e.g., about sound quality and background noise. Pointed brackets denote non-linguistic audible events like rattles (`<#Rascheln>`).

Inserted in the transcripts are also prosodic markers for boundaries and accents (see Table 11.1). Note that the system is strictly developed along the processing needs of the system. It continues work done in Verbmobil, and extends it to include especially accent information.

[B2]	weak phrase boundary
[B3 rise]	strong phrase boundary with rising intonation
[B3 cont]	strong phrase boundary with steady intonation
[B3 fall]	strong phrase boundary with falling intonation
[B9]	irregular phrase boundary
[NA]	secondary accent
[PA]	phrase accent
[EK]	emphasis/contrast

Table 11.1. Prosodic annotations.

11.5.2 Annotation of mimics and gestures

Currently under consideration are the annotation of mimics and gestures. They will be annotated differently from the text, exploiting the Quicktime facilities for time alignment. Currently, the WOZ group examines two tools for annotation, the Anvil tool developed at DFKI (see Section 2) and a commercial product for video annotation called Interact (http://www.mangold.de/index.htm).

The following audio-visual information is under consideration for annotations

- head gestures
    - rotation
    - slope backward/forward
    - side slope
- hand gestures defined functionally
    - function (e.g. "pointing", "no", "back")
    - modifier
        - reference on the projected image

        - raw morphological form (e.g. "circle with one hand")
        - begin and end coordinates
        - correlate word or phrase in the audio channel
- categories of emotional or mimic expressions
    - anger
    - boredom
    - joy/success
    - astonishment
    - neutral or undefined
    - face partially not visible (if the person moves out of the camera's visual field)
- rating of the annotation

The annotation format is currently an open question.

11.6 Current open questions

The data collection in SmartKom is evolving. First WOZ dialogs have been distributed to the partners on DVD-Rom. These data contained, besides the audio-visual data, the transliteration and the basic prosodic annotation, exploiting the Quicktime format to time align the audio-visual data with transliterations.

For the first system development and test phase resulting in a demonstrator of SmartKom at the end of 2000, this data is important. The project participants can test and evaluate the data according to their needs. They will provide feedback to the data collection site with further requirements.

Continuing from the experience, the annotation and mark-up requirements are strictly driven by the need of the system developers. It is very important that the information that is required is available. Information that might be useful in the eyes of an external observer but not in the eyes of the developers gets a lower priority.

A big problem with the first approach to annotation is how to describe the various modalities and how to link the transliterations at various levels. While for prosody there is a rich background available from Verbmobil, this is not the case for mimics and gestures.

Current open questions that are worked out in cooperation between data collection and module developers are:

1. which units are to be annotated?
2. which format to use for annotation?
3. how to link between levels?
4. how to ensure objectivity/inter-coder reliability?

For the first question, SmartKom can continue the experience in case of the audio data. For video, there are plenty of open questions.

The format for the transliteration of audio data is currently also a legacy of Verbmobil. Within the implemented system all information exchanged between modules is based on XML schemes. To enable a seamless integration of the data in the test and development flow of SmartKom, it should be considered whether the mark-up can be based on the XML schemes used in the implemented system. Figure 11.4 shows some parts of the XML gesture definition as currently used in SmartKom.

This would also give rise to a solution for the third question: XML provides mechanisms to link documents across document boundaries.

The fourth question is especially relevant for the video data. While there are established methods to write annotation manuals for speech data, the equivalent for video is not available, according to the WOZ group. Also, reports for inter-coder reliability studies are not well known –hints for this are happily accepted by the data collection group of SmartKom.

```
...
<complexType
    name="GestureEdge" content="elementOnly">
  <element
      name="recorded" type="sk:TimeInstant"/>
  <element
      name="duration" type="sk:TimeDuration"/>
  <element
      name="position" type="sk:Coordinate2D"/>
  <element
      name="type" type="sk:RawGestureType"/>
  <element
      name="kind" type="sk:RawGestureKind"/>
  <element
      name="probability" type="sk:Probability">
  </element>
</complexType>

<simpleType
    name="RawGestureType" base="sk:String">
  <enumeration
      value="tarrying"/>
  <enumeration
      value="pointing"/>
  <enumeration
      value="unknown"/>
</simpleType>

<simpleType
    name="RawGestureKind" base="sk:String">
  <enumeration
      value="static"/>
  <enumeration
      value="dynamic"/>
</simpleType>
```

Figure 11.4. Parts of the XML gesture schema, Version 0.0.

11.7 Conclusion

This contribution gave a short overview of the current state of the data collection in SmartKom.

It should be clear that for the complex task of annotation of audio-visual material, tools are needed that:

- allow for a uniform view on all media types and multiple media streams recorded
- ease transliteration
- ease inter-level linking

Additionally, what's currently missing is a concise overview of tools, annotation schemes and annotation practice for multi-media data. In contrast to, e.g., dialog act annotation, where MATE contributed a good overview on schemes, tools, and corpora, there is currently no reference available that can be used by data collection sites and system developers as a starting point.

11.8 References

Nicole Beringer. SmartKom - Datensammlung und Evaluation. LMU. SmartKom Memo 2.

Nicole Beringer. SmartKom - Systemdesign und Evaluation von Wizard-of-Oz-Aufnahmen einer multimodalen Dialogschnittstelle. LMU. SmartKom Memo 3.

Nicole Beringer. Transliterationslexikon. LMU. Februar 2000. 52 Seiten SmartKom Technical Document 2.

12 syncWRITER 2.0

12.1 Introduction

syncWRITER is a transcription and annotation tool developed for Apple Macintosh by MED-I-BIT, Hamburg, Germany, in close cooperation with the Center for German Sign Language of the University of Hamburg. It is designed to provide help for transcription and annotation of synchronous "events" such as speech and video data. The synchronous events are vertically aligned along a set of tiers; synchronization among the different parts of the tiers is ensured by means of "synchtabs" (see below). syncWRITER is a commercial product. A demo version can be freely downloaded from http://www.sign-lang.uni-hamburg.de/software/software.html. The present review is based both on general descriptions of the tool and on previous real experience.

12.2 Software design

No data is available.

12.3 Platform requirements

syncWRITER runs on any Macintosh from Macintosh Plus upward, and needs 4Mb of RAM and System 7.0 or newer. For using the QuickTime features, QuickTime 1.5 or newer must be installed. For AppleScript scripting AppleScript 1.1 or newer is required. syncWRITER is available in English and German versions.

12.4 Interface and Functionality

The following functionalities are supported:

- Conversation Transcription;
- Interlinear Translation;
- Text Annotation and Text Version Comparison;
- Synchronization of text with audio/video files.

12.4.1 Transcription

Regarding transcription, the main feature of syncWRITER is that it makes use of 'scores', comparable to those in use in musical notation: like an instrument, each role in the transcription is assigned a track, which unfolds continuously from left to right (cf. Figure 12.1). The transcription text is entered into a multi-track tape, the "document name (Tape)" window. Up to 128 tracks per document may be created and named. Tracks in syncWRITER can hold text (for example spoken words, description of gestures, comments in different tracks), pictures or a movie. Apart from the conversation transcript, tracks may be used for annotation, comments, translations, and other things. Syncwriter does not enforce any peculiar transcription formalism, allowing to adopt any transcription/annotation conventions.

In the track markers at the left edge of the tape, active tracks are highlighted by a blackened rectangle. The target position that is needed when creating new tracks or moving tracks is depicted by a triangle (either in between two tracks or above the first or below the last one). Activation of tracks or selection of the target position is done by clicking into or in between the track markers.

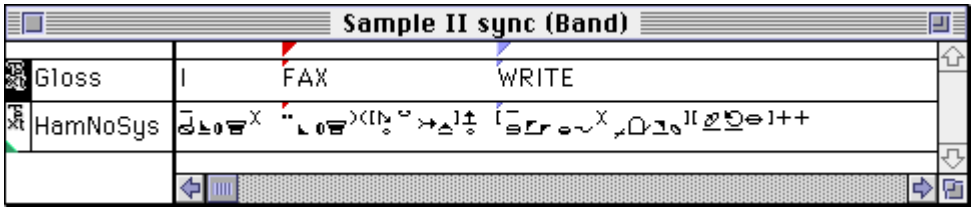


Figure 12.1. Multi-line transcription.

New text-, picture-, or movie tracks are created with the menu commands “New Text Track”, “New Picture Track”, etc. from the “Track” menu. Active tracks may be moved or deleted using commands from the “Track” menu.

Everything in one row comprises a track. There are three parts to each track: the track marker, the name field, and the content field. The speakers’ names or any arbitrary transcription headers can be entered into the name fields. The transcription text is to be input into the content fields.

Tracks are “activated” by clicking into their track markers with the mouse, before being worked upon (for example with move or delete track operations).

12.4.2 Text Entry

The name of a track may be typed into the Name Field on the left side of a track. The Content Field on the right side of a track will endlessly scroll horizontally, as more and more text is entered into it. Text attributes like font, size, style, and colour may be modified using the commands in the “Text” menu. Text selection may be created by click dragging with the mouse and whole words can be selected by double clicking. By holding the Shift key, existing selections may be enlarged.

12.4.3 Synchronizing

The simple basic idea is that transcriptions of linguistic and non-linguistic events taking place synchronously at the same time are aligned to the same horizontal position (cf. Figure 12.2). In this way, transcription can avoid those special symbols indicating speakers’ overlap, since the symbols indicating speakers are sufficient. It should be noted that this way of displaying overlap considerably increases transcripts’ readability and represents a very intuitive means of representation; on the other hand, however, it does not delimit the precise stretches of speech that overlap, only marking the starting point of an overlap and not giving information about the different speed with which two overlapping stretches may be uttered.

Text position from different tracks may be lined up using special markers or SyncTabs (synchronous tabs) to depict synchronicity. SyncTabs are displayed as small triangles above the first track. Text can be synchronized to a selected target SyncTab by setting the insertion point to the position in the text to be synchronized and choosing the "Sync To Target" menu item. This marks the position and all positions previously bound to the syncTab as simultaneous.

SyncTabs are not attached to locations on a page like tabs in traditional word processors, but they move with the text. Necessary adjustments to keep the entries of a sync in horizontal alignment are handled automatically by syncWRITER. If further text is added to the left of a SyncTab, all following SyncTabs and the text bound to them will be moved right, so no manual corrections are needed when more text is added or removed in the middle of a track.

Clicking into the text sets an insertion point. Choosing “New SyncTab” from the “Sync” menu creates a new SyncTab. A ruler above the topmost track will show the SyncTab as a triangle, while a smaller triangle in the track will show the newly created “entry” in that SyncTab. Clicking into the text of a different track and choosing the “Attach to Target SyncTab” from the “Sync” menu will make the insertion mark jump to below the SyncTab, and a small triangle will appear in this track, too.

When a second SyncTab is created, its triangle will appear red (solid black on black and white screens), and the first SyncTab will change to a light blue (black outline on black and white screens). The red SyncTab is the designated target SyncTab, and the “Attach to SyncTab” command will always attach to the target SyncTab.

It should be noted that the limited choice of colours for the SyncTabs makes syncWRITER not well suited for the purposes of multi-level annotation, in that it does not provide users with a means of differentiating between levels of synchronization of the information. However, this drawback could easily be avoided by adding a colour palette to SyncTabs.

Simply clicking on the triangles selects a target SyncTab. The small triangles that depict the entries of a SyncTab in the tracks may be shown or hidden using the “Show SyncTab Entries” command. In order to delete a SyncTab, all its entries have to be deleted by selecting the entry and pressing the Delete key. The SyncTab will disappear when its last entry is deleted.

Figure 12.2. A transcription sample with text and video synchronized.

It is not possible to line up two text positions if the text display would have to be compressed to achieve that. For example, that would happen if one tried to line up two temporally separate text positions in a conversation transcription. In such cases, if the result would be having to compress text to zero length, the “Attach” command is disabled. This would be the case if one tried to define that *a* precedes *b*, *b* precedes *c*, and *c* precedes *a*.

In all other cases syncWRITER will perform the "Attach" by inserting the exact amount of space needed to avoid compression.

In addition, as a text editor syncWRITER features extensively configurable search and replace operations. For example, it is possible to replace only in a selected set of tracks, leaving other tracks untouched. Undo operations are available for all major operations that alter the document.

Movie tracks can be added to documents in a very simple way. Directly in the program it is possible to open the movie in its own window and play it back, in full motion or frame-by-frame, to forward and backward, single-step and pause, or repeat it as often as needed. For passages difficult to analyze, suncWRITER offers a loop function.

A video can be segmented directly from the movie window while the video is being viewed. In the movie track, each segment is represented by a thumbnail which is automatically created in the segmentation process. Clicking on a thumbnail opens the play-back window showing only that specific segment.

The same range of functionalities hold for audio files.

The AppleScript support enables the automation of everything from complex analysis procedures to remote controlled construction of documents. The complete data content of a syncWRITER document may be accessed by AppleScript, so there is no limit to the application of scripts.

For printing and exporting into other applications, the interlinear transcription can be broken into blocks, resulting in a score-like format (cf. Figure 12.3). This partitioning is configurable and may be adapted to special requirements. It is further possible to export the document as a temporally ordered list of text segments, i.e. in the ‘traditional’ transcription format.

The amount of text is limited only by available memory. For printing and exporting into other applications, the interlinear transcription can be broken into blocks, resulting in a score-like format. This partitioning is configurable and may be adapted to your requirements. It is further possible to export the document in a more ‘traditional’ format, i.e. as a temporally ordered list of text segments.

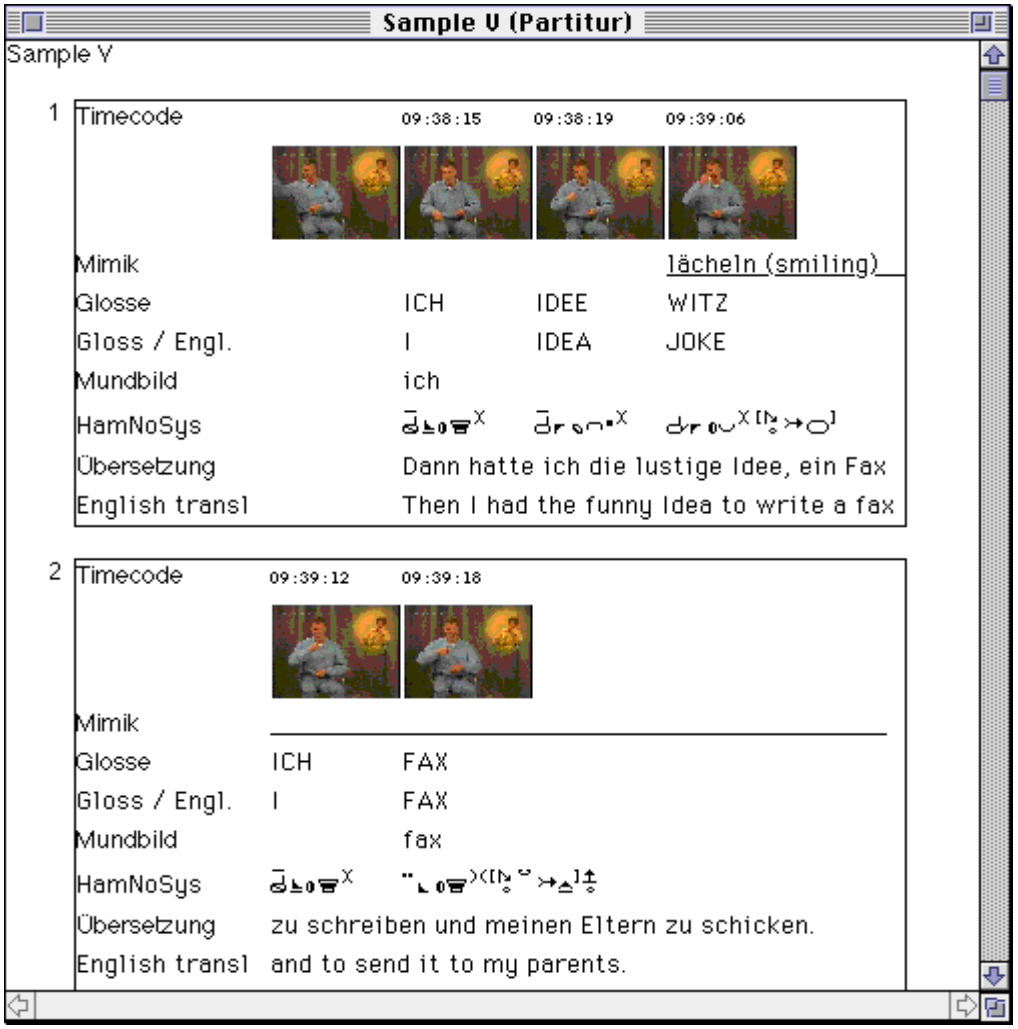


Figure 12.3. The score format for printing.

12.5 Conclusion

Due to its conceptual flexibility, syncWRITER is well suited to meet the needs of multi-modal dialogue annotation. On the other hand, one major drawback seems to be the fact that all codings and movies are necessarily contained within a single "score", and are not segmentable into independent units. The main deficiency, however, is the fact that information is only aligned to a single point in the score instead of to a segment of video, as for example in Signstream. Thus, there is no notion of an object having a certain start and end frame and duration. Consequently, there is also no temporally aligned display that shows the overlap among the extents of co-occurring objects.

12.6 References

General information about syncWRITER: http://www.sign-lang.uni-hamburg.de/software/syncWRITER/info.english.html

Hanke, T. & S. Prillwitz (1995) SyncWRITER: Integrating Video into the Transcription and Analysis of Sign Language. In H. Bos & G. Schermer (eds.), Sign Language Research 1994: Proceedings of the Fourth European Congress on Sign Language Research, Munich, September 1-3, 1994. International Studies on Sign Language and Communication of Hamburg: Signum (1995).

Papaspyrou C. & H. Zienert (1990) The SyncWRITER Computerprogramme. In S. Prillwitz & T. Vollhaber (eds.), Sign Language Research and Application, International Studies on Sign Language and Communication with the Deaf, Vol. 13, Hamburg: Signum, 275-294.

13 TalkBank

13.1 Introduction

The TalkBank project began in 1999. It is an interdisciplinary research project funded by a five-year grant from the National Science Foundation (http://www.talkbank.org/) and hosted by Carnegie Mellon University and the University of Pennsylvania. The project includes developing a number of tools. Thus this section is not a description of a single tool, but rather a description of the entire TalkBank project and the tools that it includes at the moment.

According to the original proposal of the TalkBank project, the project was launched due to the fact that most researchers studying communicative interactions are using videotape and audiotape technology with time-code generators to insert codes for alignment with transcriptions. The use of this technology means that a lot of time is spent on the rewinding of tapes and it is therefore cumbersome to go through data several times. The introduction of CD-ROM, DVD, large discs etc. makes it possible to replace this old technology with a purely digital one relying on XML, internet friendly programming languages and various video/audio decoders and encoders.

To begin with, TalkBank is focusing on five of the fifteen disciplinary groups that were mentioned in the original proposal (http://www.talkbank.org/resources/talkbank.pdf):

- Animal Communication (http://www.talkbank.org/animal/)
- Classroom Discourse (http://www.talkbank.org/class/)
- Linguistic Exploration (http://www.ldc.upenn.edu/exploration/)
- Gesture and Sign (http://www.talkbank.org/gesture/)
- Text and Discourse (http://www.talkbank.org/text/)

From the TalkBank project homepage one has access to the resources of classroom discourse (http://www.talkbank.org/class/) and linguistic exploration (http://www.ldc.upenn.edu/exploration/). One needs a username and a password to access the classroom discourse data. Should one be interested in working with the classroom discourse data, one can contact Brian MacWhinney at macw@cmu.edu, who will process the request.

All of the material at http://childes.psy.cmu.edu is being viewed as a subset of TalkBank. The CHILDES database is being converted to XML format. This database also includes hundreds of hours of audio linked to transcript. In addition LDC has released some of their CallFriend Switchboard data for TalkBank use (http://talkbank.org/data.html). Moreover, LDC and TalkBank will soon release additional segments of Jack DuBois' Corpus of Spoken American English for TalkBank.

The partners in the TalkBank project to some extent overlap with those in the ATLAS project (cf. Section 3), and there also seem to be close links between the two projects. For instance, TalkBank sponsors LDC's work on ATLAS.

13.1.1 Objectives and plans

The goal of TalkBank is to foster fundamental research in the study of human and animal communication. TalkBank will provide standards and tools for creating, searching, and publishing primary materials via networked computers.

TalkBank aims at uniting the large number of different computer based formats, standards and programs for archiving and annotating data under a common framework. In order to do so, TalkBank aims at providing tools that allow each research community to use its familiar analytic language. Furthermore, TalkBank aims at creating a distributed, web-based data archiving system for transcribed and audio data on communicative interactions.

The following types of data resources are to be included in the TalkBank project in the future: mothers talking with their children, family dinner table talk, classroom interactions, animal cries, sign language, formal debates, phone calls, talk with foreigners, club meetings etc.

TalkBank aims to cover the domains of math and science learning, conversation analysis, text and discourse, second language learning, corpus linguistics, speech production, aphasia, language

disorders and disfluency, first language acquisitions, gesture, sign language, psychiatry, conflict resolution, behavioural analyses, animal behaviour, anthropology, field linguistics, speech analysis, semi-structured data modelling, and human-computer interaction. This means that the existing and future tools of TalkBank must be able to handle dialogue, gesture, facial expressions, human-computer interaction, etc.

The idea of TalkBank is that researchers will be able to locate a particular segment of an interaction and immediately play back that section on their computer monitors. It is planned that TalkBank will facilitate comparisons across social groups, language, situations and provide tools for working in detail on single populations, as well as collaborative commentaries that test competing interpretations against a constant set of data.

TalkBank promises to provide a transcription tool, a commentary tool, alignment tools, browsing and visualisations tools, and query tools.

All the tools will be made freely available on the internet using the servers of the LDC and CHILDES projects. Subject to permission of the authors and confidentiality considerations, the Codon (see below) annotations and the primary audio and video data will also be made available both on the Internet and through CD-ROM or DVD.

13.1.2 Tools and documents

No final version of TalkBank is available at the time of writing, but the following tools and documents are being developed.

- Annotation standards (http://www.talkbank.org/annotation.html)

  This page leads to descriptions of tools and formats for creating and managing *linguistic annotations*. The term 'linguistic annotation' is set to cover any descriptive or analytic notations applied to raw language data. The focus of the page on annotation standards is on tools which have been widely used for constructing annotated linguistic databases, and on formats commonly adopted by such tools and databases. This facility has not been created exclusively for the TalkBank project. The homepage is updated on demand, and was last updated in March 2000 (time of writing is December 2000).

- Streaming video and audio for either Macintosh or Windows.

  (http://www.talkbank.org/stream/winstream.html)
  (http://www.talkbank.org/stream/macstream.html)

  This facility should be working on both Windows and Macintosh. We tried to install it on a windows machine but never managed to make it work and thus have not been able to test it. The installation of the different software that is required for this facility is a very extensive task, which requires a fast internet connection and a good amount of time. The facility relies on CLAN (cf. Section 4), QuickTime and Windows Media Player, which are all software products that were developed for other purposes than the TalkBank project. The data examples that can be found on the web pages also originate from other projects.

- An illustration of how to make direct links to video and audio from an HTML web page (http://www.talkbank.org/stream/htmlplay.html.) This facility relies on JavaScript and uses time markers created in CLAN and then converted to QuickTime format. It only works reliably on Windows platforms using Internet Explorer 5.

  The web page leads to a list of different sound or video examples of data. Having tested the possibilities on the page it is understood that by "illustration" the TalkBank people mean examples of data transcriptions with links to their respective sound and/or video sources and not an explanation of how one makes such direct links.

- Versions of the Transcriber (http://www.ldc.upenn.edu/mirror/Transcriber/) software that facilitate analyses for projects ranging from ethnology to language acquisition.

  Transcriber (http://www.etca.fr/CTA/gip/Projets/Transcriber/) is a tool for assisting the creation of speech corpora. It was originally developed in France at Centre Technique d'Arcueil (http://www.etca.fr/CTA/) in collaboration with LDC for transcribing a corpus of Broadcast News material in French. It allows to manually segment, label and transcribe speech signals, for later use

in automatic speech processing. It is more specifically geared towards the transcription of long duration broadcast news recordings, with labelling of speech turns and topic changes, but can also be used with other types of corpora. It provides a user-friendly interface which is configurable.

- The CLAN program, cf. Section 4. CLAN was developed long before TalkBank started but is now also considered part of this project. CLAN represents an integrated set of analysis schemes and multimedia players that sets the lower level of functionality for the new tools being built in TalkBank. In particular, CLAN has some powerful features of multimedia transcription. Although the Transcriber interface is prettier, the video and audio CLAN functionality should be more extensive from the transcribers point of view. In extending these functions over the last year the TalkBank people have often relied on creative ideas developed in Transcriber.
- An adaptation of the CLAN software (http://childes.psy.cmu.edu/mac/clan.sit) that permits direct editing of digital video on miniDV casettes over FireWire connections to Macintosh. (This facility does not work yet)
- A pair of demo tools (http://talkbank.org/tech/dtds/), which at the moment only run under Unix including Linux. These demo tools should illustrate the use of the AIF (The ATLAS Interchange Format), cf. Section 3. However, the demo tools could not be found at the time of writing (December 2000).

Furthermore, an XML-based annotation system called Codon is being developed in order to serve as the formal specification for data in TalkBank. The development of Codon includes the development of tools that can facilitate the entry of new and existing data into the Codon format, link transcriptions to speech and video, and support collaborative commentary from competing perspectives. Codon is to serve as an abstraction layer under which different annotation standards can be used. The aim is that all current transcription systems can be translated into Codon and that Codon will facilitate the direct comparison of complementary and competing analyses of a given dataset. Bird and Liberman's annotation graphs will be used as a specification of the temporal alignment of linguistic units, but the intention is not that annotation graphs should replace standards and tools which have been accepted by existing communities, but rather to make the different descriptive and analytical practices, formats, data and tools from these communities available to all of the communities, by using annotation graphs as an accepted standard interlingua understandable to all. In addition, Codon will be XML compatible and should be able to integrate with tools for web-based information retrieval from databases.

The annotation system Codon is planned to solve the following tasks: foster the creation and adoption of open standards for all levels of annotation of communicative structure, including marking of movements, orientation, gesture, orthography, part-of-speech tagging, syntactic structure, syntactic class, co-reference, phonetic segmentation, overlaps, disfluencies, code-switching, prosody, facial expressions, situational background and participant identity.

Apart from CLAN which is described in Section 4, it has only been possible to test Transcriber among the tools mentioned above. The version tested is 1.4.2.

13.2 Software design

TalkBank will at some point in the future consist of a number of tools as mentioned above. The main consideration is that these tools should be platform independent. There is no specification of a date on which all the tools are expected to be up and running.

Transcriber is developed using the scripting language Tcl/Tk and C extensions. It relies on the Snack sound extension, which allows support for most common audio formats, and tcLex lexer generator. Within the next year the TalkBank project plan to switch from Tcl/Tk to Python. Transcriber is distributed as free software under the GNU General Public License.

The implementation language for the Codon editor will be Tcl/Tk or Java. The CHILDES editor (CED) will be a starting point for the design. CED provides the user with a text editor and a systematic way of entering user-determined codes. In the coding mode, CED allows the user to establish a predetermined set of codes and then to march through the file line by line making simple keystroke movements that enter the correct codes for each utterance selected. Once a file has been fully coded in CED a variety of additional analyses become possible. On Macintosh the CED editor allows the transcriber direct access to digitised audio records.

There will not be a single monolithic database. Instead a consortium of databases will apply to the Codon standard and can thus be manipulated with Codon tools.

A commentary tool that makes it possible to cross-annotate existing annotations for comparison reasons is also planned.

A number of existing alignment tools for aligning textual transcripts with audio data will be adapted to the Codon format.

13.3 Platform requirements

Transcriber runs on Windows, Mac and Unix platforms. The needed Media Player and Quick Time for streaming video and audio should work on Macintosh and Windows but we have not been able to run the streaming video and audio neither on Windows 2000 nor on NT.

To run the streaming video and audio on Windows one needs to install

- Windows Media Player 6.4: http://www.microsoft.com/windows/windowsmedia/en/download/default.asp
- A recent version of Quick Time for Windows: http://www.apple.com/quicktime/download/
- A current version of Windows CLAN: http://childes.psy.cmu.edu/html/ca.html

To run the streaming video and audio on Macintosh one needs to install

- Version 4.1.1 of QuickTime from http://www.apple.com/quicktime/download/
- A current version of Macintosh CLAN: http://childes.psy.cmu.edu/html/ca.html

In order to run some of the tools of TalkBank a soundcard and equipment capable of viewing streaming video and audio (which includes a fast internet connection) is required.

13.4 Interface

Only Transcriber is described in the following since this is the only tool – apart from CLAN – we were able to test.

Figure 13.1 shows the window which appears when Transcriber is opened. In the "Open transcription file" box one can choose to open one of the demos of transcriptions that come with the downloaded program. Or one can create a new file by simply closing the "Open transcription file" box. One may also open a new or existing transcription file by pressing "new trans..." or "open trans..." in the FILE menu. Transcriber can open transcription files of the following formats. *.trs, *.xml, *.cha, and *.typ.

As can be seen it has the "usual" Windows program format with some extras, such as "Signal", "Segmentation" and "Options". "Signal" has to do with the playing mode of the sound, "Segmentation" with the transcription and "Options" with how one wants a transcription to be displayed.

The "Help" function is very extensive, and takes one through every step of the program one may wish to explore. It is divided into the following four groups:

- Presentation
- Main features
- Interface
- User manual

The presentation section provides a very brief overview of system capabilities, development issues, platform requirements, and availability. Main features are described in Section 13.5. The description of the interface uses the French names for "File", "Edit", etc. The User Manual is very user friendly and easy to understand.

Figure 13.2 shows a transcription in progress. The transcription can be found and handled at the top of the window and the sound can be managed at the bottom. Transcriber can read the following kinds of sound files: *.au, *.wav, *.snd, *.sph, *.sig, *.smp, *.aif, *.aiff, and *.mp3

The user can configure colour, font, command buttons etc. The program may seem at bit overwhelming to new users. However, due to the extensive "Help" function this problem can be overcome if one takes the time to read through how the program works.

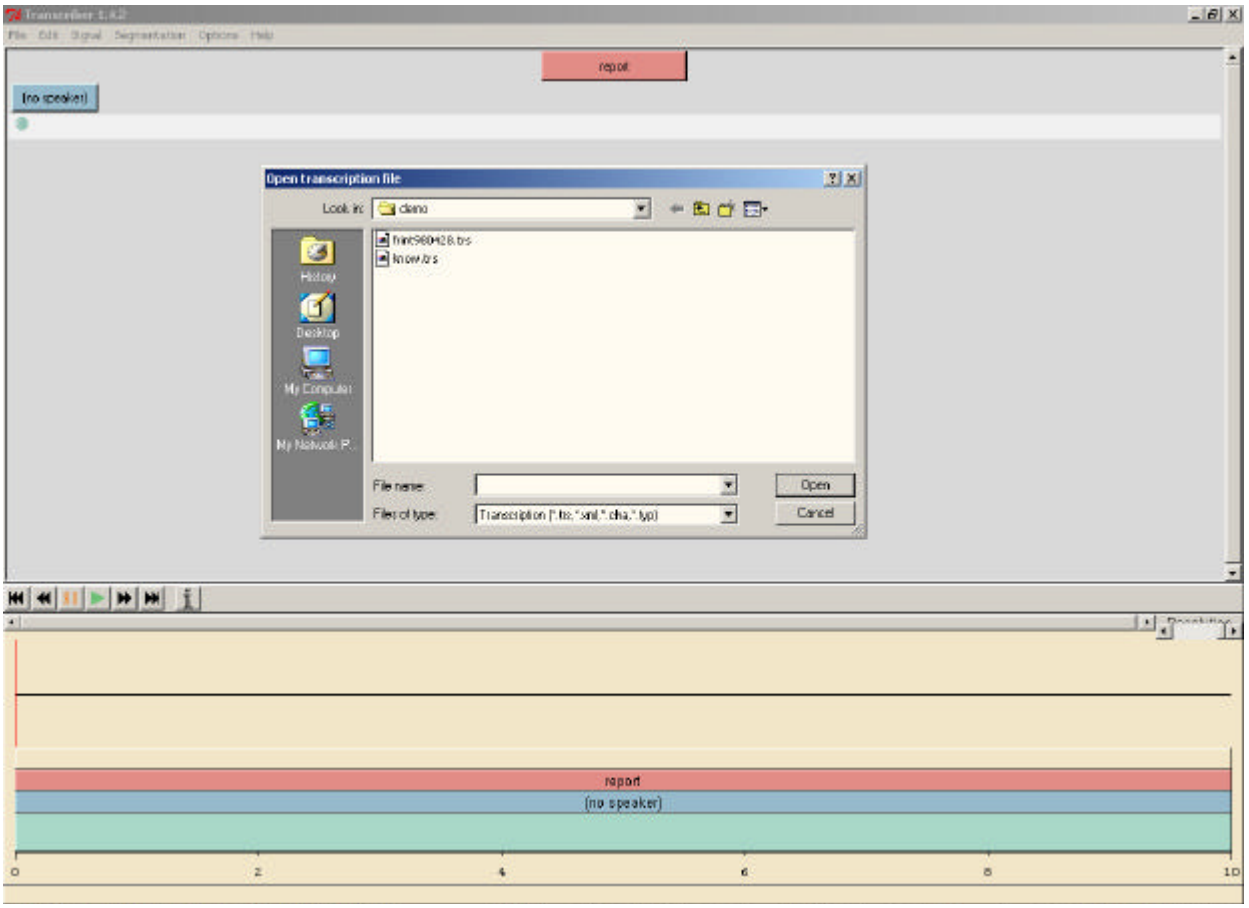


Figure 13.1. The Transcriber window.

13.5 Functionality

The main features of Transcriber are described below.

When transcribing one can:

- manage several layers of segmentation: basic segmentation for orthographic transcription (e.g. at each sentence or at each breathing), speech turn segmentation (new speaker) and section segmentation (new topic); these 3 layers are hierarchically embedded. A fourth segmentation in background acoustic conditions is also enabled;
- display segmentations under the signal and in the text editor;
- synchronize the cursors in the text editor window and in the signal window: as soon as the cursor moves in one window, it is moved in the other one and appears within the same segment;
- manage a speaker list with a description of each of them (name, type, accent) and further modification of its characteristics;
- in the same way, manage a list of topics, find sections about a topic and import topics from another file;
- when opening a file, the automatic search for the corresponding sound file begins; if none is found, the user is asked to locate it.

When handling sound Transcriber can:

- handle long duration audio files (up to several hours)
- handle several playback modes through keyboard combinations: play/pause, play the current segment, the selection, around the cursor.

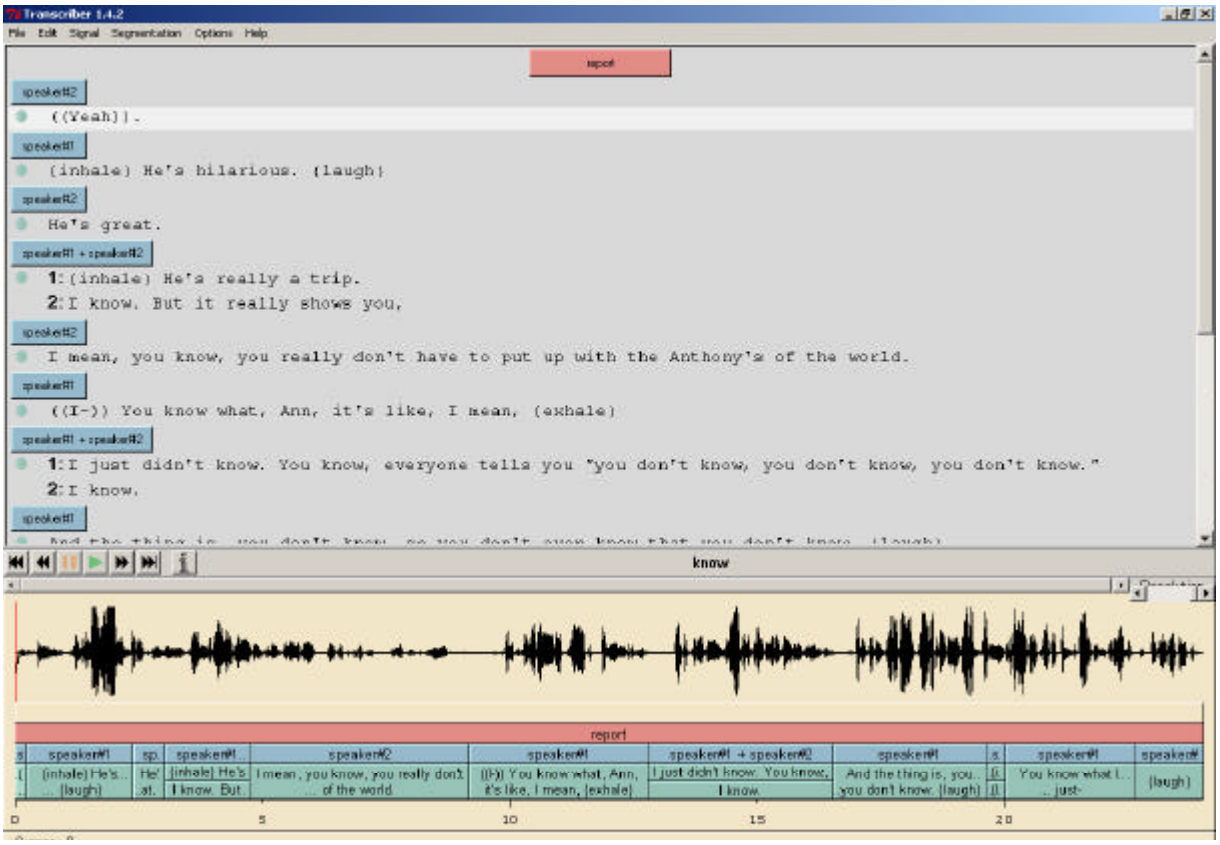


Figure 13.2. Transcription.

Other features of Transcriber:

- display cursor position (also during playback), selection width and other messages under the signal;
- monitoring of user's productivity during the session;
- information window about the signal (filename, duration, format), and about the transcription (number of segments, words);
- one can configure colour, font, command button etc.

13.6 Conclusion

Talkbank is a recently started project so there is not much to evaluate yet in terms of tools produced by TalkBank. It seems that what is available at the moment originally derives from other projects. For instance, the Transcriber Tool which we tested was originally developed in France at Centre Technique d'Arcueil in collaboration with LDC.

As regards ideas and plans Talkbank would seem to be a very ambitious project. The main idea in TalkBank– that it should be possible to use the same tool on top of existing tools and annotation standards - is good. However, only time can show what the project will actually manage to achieve and if the plans can be realised.

13.7 References

TalkBank: http://www.talkbank.org/

Transcriber: http://www.ldc.upenn.edu/mirror/Transcriber/

CLAN /Computerized Language ANalysis): http://childes.psy.cmu.edu/

Childes (the Child Language Data Exchange System): http://childes.psy.cmu.edu/

14 User needs and standards

This chapter summarises key information from the previous twelve tool review sections, compares the reviewed tools and discusses user needs and standardisation efforts on this background. The extracted key information is presented in tables to provide an overview at a glance and facilitate comparison of the reviewed tools.

14.1 Overview and comparison of reviewed tools

In the following, Tables 14.1-14.5 provide different overviews of information about the twelve reviewed tools. Each overview is briefly discussed and the information about tools compared.

Table 14.1 lists sites and countries involved in the development of a particular tool and the web address where the tool or information about the tool can be found. Seven of the twelve tools are developed by one site only. Two of these tools are German (Anvil and SyncWriter), two are from the Netherlands (MPI tools and The Observer), one is Swedish (Multitool), and two are from the USA (CLAN and SignStream). For the CSLU Toolkit, two sites are mentioned due to the fact that some of the key people involved moved from OGI to Colorado recently. So this tool could also be counted as developed by one site only. The four remaining tools are each developed cooperatively by several sites. The two American collaborative projects (ATLAS and TalkBank) are at the stage of specification and tools developed by those projects are not yet available. However, some tools originating from other projects (e.g. CLAN and Transcriber) are now considered part of the TalkBank project. The German SmartKom project is also new and contrary to the other tools and projects reviewed in this report, SmartKom is rather a user of and requirements provider for annotation tools than an annotation tool development project itself. The MATE project is a European collaborative project which has produced a workbench that is now freely available, including the source code.

Table 14.2 provides an overview of the programming language(s) used for implementation of each tool, platform requirements for using the tool, and the conditions under which the tool can be obtained. Java, Tcl/Tk and C or C++ seem to be the preferred programming languages, and XML is in many cases used for file representation. The majority of the tools run on a Windows platform. Many of these tools (at least the Java-based ones) should also be able to run under Linux and Unix (but not on MacOS). In a few cases a tool is developed to run only on a Macintosh platform. In nearly all cases an executable version is freely available. For the commercial tools the free version is typically a limited demo. In four cases (the CSLU Toolkit, the MATE Workbench, Multitool, and Transcriber (TalkBank)) we found that also the source code is freely available.

Some of the reviewed tools are viewed rather as a set of individual tools than as one tool. Table 14.3 provides an overview of whether a tool is seen as a collection of tools, in which case these are listed, or whether it is considered one tool, in which case the name of the tool is simply repeated in the second column. The third column lists the tool version (if any) we tried to test. As can be seen from the fourth column, testing caused problems in a number of cases and clearly reflects the fact that we were often dealing with unstable research prototypes. In four cases (CSLU Toolkit, MPI tools, Multitool, part of TalkBank) we had severe problems with (part of) the software in spite of the developers of the tools being very kind and trying to help. The solution then typically was to either visit the development site or communicate with the developers via email or telephone to get the information we needed. Communicating with the developers to get detailed information and screen shots is of course not as good as testing the tool oneself but still is much better than only reading about the tool in literature or on the web. In two cases the reviews have been made by the developer of the tool or by a person deeply involved in the project (Anvil and the MATE workbench). Both are research prototypes. The Anvil developer has not received any reports on testing problems from people who tried the tool. The tool is small but should be fairly stable. The MATE workbench is a much larger tool and some people have reported on problems in using the tool. A helpdesk is available at http://mate.nis.sdu.dk from which the MATE workbench developers can be contacted. In two cases (ATLAS and SmartKom) we did not test anything. Finally in five cases (CLAN, The Observer, SignStream, SyncWriter, and Transcriber (TalkBank)) there were no problems in particular to report. The two commercial tools in our review (The Observer and SyncWriter) are both in this group. CLAN

has been developed for the analysis of data transcribed in CHAT. It has been around for quite some time and is fairly well-tested by its large user group. SignStream has also been developed for a specific purpose, i.e. for research in sign languages. Transcriber is specialised to speech transcription.

Table 14.4 presents the natural interactivity and multimodality (NIMM) aspect(s) addressed by each tool in terms of speech, text, gesture and/or facial expression. It also briefly describes the main functionalities offered by each tool. One tool focuses entirely on speech (Transcriber (TalkBank)). One tool concentrates on speech and text (the MATE Workbench). One tool handles speech, text and gesture (CLAN). Four tools address speech and gesture (Anvil, the MPI tools, Multitool and SyncWriter). One tool focuses on speech, gesture and facial expression (SignStream). One tool handles speech, facial expression and text-to-speech (CSLU Toolkit). One tool addresses gesture and facial expression (The Observer). Finally, for two of the twelve tools considered (ATLAS and SmartKom) there was no software to review. Thus speech seems to be the most central element addressed by all the mentioned tools but one, i.e. by nine tools. Gesture is addressed by seven of the ten tools for which there was software to review. Facial expression is addressed by only three of the tools. The MATE workbench is by far the most advanced as regards markup of spoken dialogue. It can in principle handle any annotation level and comes with a number of example coding schemes for different annotation levels, such as dialogue acts and coreference. The other tools handling speech do either not go beyond the transcription level or at most offer annotation of e.g. dialogue acts according to a built-in anotation scheme.

Most of the tools addressing gesture have a number of basic functionalities in common including the possibility of viewing the video, adding markup, making synchronisation, making time alignment, and extracting information. Considering the actual implementations one of the most advanced and stable tools for gesture annotation would seem to be the Observer.

The Observer also includes support for annotation of facial expression. Among the reviewed tools only two other tools (SignStream and the CSLU Toolkit) claim to support annotation of facial expression. For the CSLU Toolkit the annotation support is meant for output generation in terms of an animated speaking face. Several of the tools handling gesture could probably –with more or less effort – be extended to handle markup of facial expression as well.

Table 14.5 summarises the objectives of each tool/project. Two tools were made for commercial purposes (The Observer and SyncWriter) whereas the other ten tools/projects are research tools/projects. SmartKom is in fact rather a user of annotation tools than a provider. Two tools were constructed for very specific project purposes: Anvil (a PhD project) and CLAN (created as tool support for the CHILDES project, thus based on the formats used in CHILDES and constructed to serve the needs of the (very large) project community). It might be possible to use these tools in other contexts than those they were originally built for. SignStream was developed for sign language research but can probably be generalised to cover other areas. The CSLU Toolkit contains several tools but annotation is not a principal application. The annotation functionality included is intended for the creation of multimodal speech and facial expression output which in turn is meant for pedagogical purposes. Thus the toolkit does not capitalize on annotation of large amounts of dialogues nor on query. The MPI tools and Multitool are both intended to be general purpose tools not aimed at one particular task. This is also the case for the MATE workbench and the intended results of the ATLAS and TalkBank projects. MATE, ATLAS and TalkBank, claim, in addition, to work towards standardisation.

Tool	Developer(s)	URL
Anvil	DFKI, Germany	www.dfki.de/~kipp/anvil
ATLAS	NIST, USA, LDC, USA, MITRE, USA	www.itl.nist.gov/iaui/894.01/atlas
CLAN	Carnegie Mellon University, USA	childes.psy.cmu.edu
CSLU Toolkit	CSLU at OGI, USA, CSLR at University of Colorado, USA	cslu.cse.ogi.edu/toolkit/
MATE Workbench	NISLab, Denmark, CSELT, Italy, DFE, Spain, DFKI, Germany, HCRC, UK, ILC, Italy, IMS, Germany, TID, Spain	mate.nis.sdu.dk
MPI tools (EUDIC0 and CAVA)	Max Planck Institute for Psycholinguistics, Nijmegen, The Netherlands	www.mpi.nl/world/tg/CAVA/CAVA.html www.mpi.nl/world/tg/lapp/eudico/eudico.html
MultiTool	Gothenburg University, Sweden	www.ling.gu.se/multitool
The Observer	Noldus, The Netherlands	www.noldus.com/products/index.html?observer/index
SignStream	Boston University, USA	web.bu.edu/asllrp/SignStream
SmartKom	DFKI, EML, Universität Erlangen-Nürnberg, ICSI, Universität Stuttgart, Universität München, MediaInterface, Philips, Siemens, Sony	www.smartkom.org
SyncWriter	MED-I-BIT, Germany	www.sign-lang.uni-hamburg.de/software/software.ht ml
TalkBank	Carnegie Mellon University, USA, University of Pennsylvania, USA	www.talkbank.org

Table 14.1. Sites involved in developing each of the reviewed tools and the web addresses where information about the tools and in most cases also a (demo) version can be found.

Tool	Implementation language(s)	Platform requirements	Availability
Anvil	Java, JMF, xml4j XML for file representation	Any platform on which Java 1.2 and JMF 2.0 can run. Tested on Windows	Free for research purposes. To download it contact kipp@dfki.de
ATLAS	Expected to be Java, C++, Tcl, XML for file representation	Not clear	Expected to be free
CLAN	C	Macintosh, Windows	Free
CSLU Toolkit	C, Tcl/Tk C++/Scheme (Festival) XML (Sable)	Windows	Free for non-commercial use; source code included
MATE Workbench	Java, XML for internal file representation	Any platform for which Java 1.2 or newer is available. Tested on Windows and Unix.	Free. Source code available under the GNU LGPL license
MPI tools (EUDIC0 and CAVA)	QuickTime based Macintosh application (CAVA MediaTagger) Java, JMF (EUDICO)	Windows (CAVA transcription editor (TED)) Macintosh PowerPC (CAVA MediaTagger) Windows, Unix and any platform that supports JMF (soon also Linux) (EUDICO)	TED – availability unkown MediaTagger – free upon request for research EUDICO – not yet publicly available
MultiTool	Java, JMF	Any platform on which Java 1.2, JMF 1.1 can run Tested on Windows	Free. Source code available
The Observer	C++	Windows	Commercial, demo version free
SignStream	QuickTime based Macintosh application	MacOS 7.1 or later, QuickTime 2.1 or later.	Demo version free for 30 days. Distributed on a non-profit basis for non-commercial use
SmartKom	Various implementation languages. XML for file representation	Linux or Windows NT	Available to project partners only
SyncWriter	QuickTime based Macintosh application	MacOS 7.0 or later, QuickTime 1.5 or later, AppleScript 1.1 or newer	Commercial, demo version free
TalkBank	Expected to be Tcl/Tk or Java (CODON Editor) Tcl/Tk (Transcriber)	Not clear Windows, Unix, and Mac	Will be free Free. Source code available under GNU GPL License

Table 14.2. Overview of implementation language(s) used for each tool, platform requirements for running the tool, and availability of the tool. Free means that an executable version is freely available.

Tool	Tools developed so far	Tool version tested	Problem with test version
Anvil	Anvil annotation tool	Version 2.2	None in particular
ATLAS	None	N/A	N/A
CLAN	CLAN Editor	Last version available on the web site	No problems to report
CSLU Toolkit	RAD, BaldiSync, SpeechView, OGIsable, speech recognition tools, and a programming environment (CSLUsh)	Version 2.0b2	OGIsable's Tcl code seemed buggy – needed some tweaking to run. Others tools seemed to work fine
MATE Workbench	MATE Workbench	Version 0.17 from August 2000	No problems in particular
MPI tools (EUDIC0 and CAVA)	CAVA transcription editor (TED) CAVA MediaTagger Only a read-only demo of EUDICO is available	MediaTagger 3.0b from April 2000	MediaTagger worked fine. The online demo of EUDICO could not be made to run, so the reviewer had to visit MPI to test it
MultiTool	MultiTool	Beta version, but no hands-on experience	The version available on the web could not be made to work, so the review is based on personal communication with the developers. The tool is very unstable for the moment according to the developers
The Observer	The Observer (commercial tool in different versions for different purposes and platforms)	Observer Basic for Windows, version 3.0	No problems to report
SignStream	SignStream	SignStream version 2.0	No problems in particular with SignStream
SmartKom	None	N/A	N/A
SyncWriter	SyncWriter	SyncWriter version 2.0	No problems in particular
TalkBank	Only further developments of tools from other projects so far	Transcriber version 1.4.2 Support for the streaming playback of audio and video directly from transcripts, but no hands-on experience CLAN, cf. above	No problems in particular with Transcriber The support for streaming playback available on the web could not be installed correctly and made to work

Table 14.3. Overview of individual tool(s) developed so far for each tool (or toolset) reviewed. For each tool it is indicated which version was tested (if any), if it was not tested directly with hands-on experience, and whether there were problems installing and running the tool.

Tool	NIMM aspect(s) addressed by tool	Main functionalities
Anvil	Speech and gesture	Annotation tool for video files. It allows the encoding of nonverbal behaviour (e.g. gesture) and speech related concepts (e.g. dialogue acts) on multiple layers, so-called tracks and their visualization in temporal alignment.
ATLAS	No tool available but the project focuses on support for annotation of all kinds of linguistic signals	ATLAS aims at acting as an interlingua for language corpora and at facilitating the exchange and reuse of existing language resources
CLAN	Text, speech and gesture	Tool designed specifically to analyse data transcribed in the format of the Child Language Data Exchange System (CHILDES). CLAN allows the user to perform a large number of automatic analyses on transcript data. The analyses include frequency counts, word searches, co-occurrence analyses, mean length of utterance (MLU) counts, interactional analyses, text changes, and morphosyntactic analysis.
CSLU Toolkit	Speech, TTS, facial expression	SpeechView allows user-defined time-aligned labelling of waveforms, such as phonetic and higher-level transcription. OGIsable enables the user to annotate a text with properties such as voice type, facial expression, and pitch range before it is spoken by an animated agent via a TTS system.
MATE Workbench	Speech and text	Tool in support of annotation of spoken dialogue corpora at multiple levels and information extraction from annotated corpora (query). Users may use built-in coding schemes or add new ones. Converters from/to MATE XML format can easily be added. Examples are provided.
MPI tools (EUDIC0 and CAVA)	Speech and gesture	TED (CAVA) provides functions for creating time-aligned transcriptions and annotations of video tape recordings. MediaTagger (CAVA) has a number of functions for creating annotation tiers with time-aligned tags for digital video transcriptions. EUDICO includes, in prerelease implementation, some MediaTagger functionality plus additional functionality, including coordinated transcription displays and complex search functions, in various modes of operation, from stand-alone to full client-server mode.
MultiTool	Speech and gesture	Multitool includes support for synchronisation of text transcriptions, audio and video, different presentations, and a search facility for audio and video.
The Observer	Gesture and facial expression	Tool in support of recording and annotating observational data, and making them immediately available for quantitative analyses. It can be used to record activities, postures, movements, positions, facial expressions, social interactions or any other aspect of human or animal behaviour as time series of tagged data.
SignStream	Speech, gesture and	Tool for browsing and annotating linguistic data captured on

	facial expression	video, according to simultaneous layers of description. A search functionality is also included.
SmartKom	No tool available but the project focuses on speech, gesture, mimics and biometrics	N/A
SyncWriter	Speech and gesture	Tool in support of transcription and annotation of synchronous "events" such as speech and video data.
TalkBank	Speech (Transcriber) Text, speech and gesture (CLAN, cf. above)	Transcriber is a tool in support of transcription from audio files.

Table 14.4. NIMM aspects addressed by each tool, i.e. speech, text, gesture, and/or facial expression, and the main functionalities offered by the tool.

Tool	Objectives
Anvil	Anvil has been developed as part of a PhD project. The objective of the PhD project was to develop a methodology starting from an analysis of individual behaviour (using the Anvil system) and bridging the gap to computer animation with a rule-based generator that produces "life-like" gesture and posture from an input of "structured" speech.
ATLAS	The ATLAS research project has no tool ready yet. The aim is to develop a general architecture for annotation including a logical data format, an API and tool-set, and a persistent data representation. The architecture needs to be modular, flexible and extensible and facilitate the exchange and reuse of existing language resources. ATLAS also aims at acting as an interlingua for language corpora. The final goal of ATLAS is that it will serve as a conduit to enable a greater flow of language resources throughout the language research community.
CLAN	CLAN has been developed for research purposes and with the specific aim of analysing data transcribed in the format of the Child Language Data Exchange System (CHILDES). The automatic analyses enabled include frequency counts, word searches, co-occurrence analyses, mean length of utterance (MLU) counts, interaction analyses, text changes, and morphosyntactic analysis.
CSLU Toolkit	The CSLU Toolkit is a toolbox developed for research and educational purposes in the area of speech and human-computer interaction. The core tool SpeechView provides basic functionality for displaying, editing and labelling speech waveforms for input to the Festival TTS engine. The auxiliary annotation GUI OGIsable provides a convenient user interface to the Sable TTS markup.
MATE Workbench	The aim of the MATE research project has been to facilitate the use and reuse of spoken language resources by addressing theoretical issues and implementing practical solutions. The MATE markup framework has been proposed as a standard which can facilitate uniform description of coding schemes across annotation levels. The MATE workbench supports the framework, incorporates a number of MATE best practice coding schemes for different levels, and enables users to add new coding schemes for new or existing levels.
MPI tools (EUDIC0 and CAVA)	The MPI research tools are intended to support the scientific exploitation of multimedia (specifically audiovisual) corpora by linguists, anthropologists, psychologists and other researchers by providing convenient user interfaces and functionality for AV corpus transcription, annotation of videos, querying (upcoming) and, eventually, corpus metadata browsing (upcoming).
MultiTool	Multitool has been developed as part of a research project. It is an attempt to build a general tool for linguistic annotation and transcription of dialogues, browsing, searching and counting. An objective of the tool has been that it should be able to handle any number of participants, overlapping speech, hierarchical coding schemes, discontinuous coding intervals, relations and synchronisation between codings and the media file. The target users are linguistics researchers both from university and industry who wish to analyse spoken language in all its facets, ranging from visual to statistical information.
The Observer	The Observer is a professional commercial system for the collection, analysis, presentation and management of observational data. It can be used to record activities, postures, movements, positions, facial expressions, social interactions or any other aspect of human or animal behaviour as time series of tagged data. The tool has been designed in order to help people to record and annotate observational data, making it immediately available for quantitative analysis. The data can immediately be subjected to statistical analysis through recourse to a built-in data analysis module, in order to test an experimental hypothesis. The

	Observer comes under three standard system configurations: The Observer Basic is for data collection with a stand-alone computer; The Observer Mobile, for data collection with a handheld computer, and The Observer Video-Pro, for data collection from video.
SignStream	SignStream is a database tool for analysis of linguistic data captured on video, developed as part of the American Sign Language Linguistic Research Project. The tool was originally intended to be applied in research on Sign Languages, where video is the primary medium for collecting data, to overcome the limitations of conventional representations of signed language data in written gloss form. In spite of this, SignStream can in principle be used for any kind of linguistic research that relies on video data, including language acquisition, bilingualism, neurolinguistics and language disorders, phonetics and phonology, sociolinguistics, discourse and narrative studies, pragmatics, analysis of the gestural component of spoken languages; and computational linguistics.
SmartKom	SmartKom's goal is research in the area of multimodal interaction, leading to the development of a self-explaining, user adaptive interface to various applications. The project is guided by the aim to merge the advantages of dialogue-based communication with the advantages of a mixture of graphical user interfaces and gesture and mimic interaction. SmartKom develops core functionalities for intelligent communication assistants which analyse language, gesture and miming. The assistants interpret these input modalities in the context of the dialogue and initiate actions accordingly. For training, test, and evaluation purposes data must be collected and annotated.
SyncWriter	syncWRITER is a commercial transcription and annotation tool developed for Apple Macintosh by MED-I-BIT, Hamburg, Germany, in close cooperation with the Center for German Sign Language of the University of Hamburg. It is designed to provide help for transcription and annotation of synchronous "events" such as speech and video data.
TalkBank	The objective of the TalkBank research project is to provide standards and tools for creating, searching, and publishing primary materials via networked computers. The project aims at uniting the large number of different computer based formats, standards and programs for archiving and annotating data under a common framework. In order to do so TalkBank aims at providing tools that allow each research community to use its familiar analytic language. Furthermore TalkBank aims at creating a distributed, web-based data archiving system for transcribed and audio data on communicative interactions. TalkBank tools must be able to handle both spoken dialogue, gesture, and facial expressions. TalkBank promises to provide a transcription tool, a commentary tool, alignment tools, browsing and visualisations tools, and query tools.

Table 14.5. Objectives of tools and/or projects.

14.2 User needs and standardisation efforts

The development of commercial and research NIMM systems is a growing business today. To develop any such system, NIMM data resources are needed for many different purposes, including information gathering, research, training and evaluation. It is a well-known fact that data resources are expensive to produce and exploit, and so is the development of tools which may facilitate and reduce the cost of production and exploitation of data resources. Thus it would seem highly attractive to build general tools which can be reused and shared by a much larger community than a single research lab or project. It would of course also be desirable to produce reusable data resources. However, in many cases the data resources needed are specific to the project or system under development and have to be produced for that particular purpose.

The basic purpose of a general tool would be to support transcription and annotation of NIMM data at multiple levels and across levels and modalities, and to enable information extraction from the annotated data. The demand for such a tool is likely to increase – both in industry and academia - along with the increasingly advanced NIMM systems we want to build.

A tool can make the transcription, annotation and data analysis process much more efficient compared to a completely manual process. Without a tool, all data has to be hand-coded and information must be extracted manually, which is a slow and error-prone process. Computers are much better and faster than humans are at counting and at quickly presenting extracted data from different perspectives.

The basic need for a general tool is strongly reflected in the reviewed tools many of which are intended to be general rather than specifically supporting one particular project's needs. However, it is also clear that it is not easy to build well-functioning general tools. Moreover, tools resulting from research projects will often be research demos with what that entails in terms of fragile and buggy software.

However, the increasing interest in the NIMM area is likely to lead to a larger market for NIMM annotation and data analysis tools and thus also to a commercial interest in such tools. Two of the reviewed tools are in fact commercial tools for limited parts of the NIMM area, so some commercial interest is already evident. If industry becomes more heavily involved – also in research projects - in general tools development for NIMM data annotation and analysis, it is likely that the resulting software will be more stable and somewhat closer to a commercial product than the majority of tools reviewed in this report.

The reviewed tools reflect a good many user needs. It is clear that users have many different needs and that the NIMM area is a large area. If one tries to summarise the most important overall needs which a general tool in support of NIMM data transcription, annotation and information extraction should satisfy, such a tool would probably need to fulfil at least the following specification:

1. Flexible and open architecture which allows for easy extensions and addition of new tool components;
2. Separation of user interface from application logic (internal representation) and data (external representation) so that each can be changed independently;
3. Transcription support;
4. Annotation support at different levels, of different modalities, and of the interaction across levels and modalities;
5. Powerful functions for query, retrieval and extraction of data from annotated corpora; tools for data analysis, including statistics;
6. Adequate data presentation (visualisation); possibly synchronised view of different layers of annotation and of different modalities.
7. Easy-to-use interface;
8. Support for easy addition of new coding schemes and for defining new visualisations;
9. Possibility of reusing existing data resources via conversion tools.

Most importantly, perhaps, such a tool (or toolset) has to be robust, stable and work in real time.

Other comments on the above points are:

(1) A general tool produced by a research project or a company is not likely to cover the needs of all users. Thus it is important that users can make their own additions to the tool and that APIs which support the addition of new tool components are made available.

(2) External data representation should be separated from the user interface via an intermediate logical layer so that the former two layers can be changed separately without influencing one another. Good GUI software practice prescribes a three-layered structure as a minimum: the user interface, an intermediate (logical) layer, and a data representation layer. Often these layers will be further subdivided. Thus, the intermediate layer proposed by ATLAS is not an innovation as seen from a software perspective, but it would certainly be desirable to have this layer also in annotation tools where it seems to have been used much too little. MATE also has the idea of an intermediate layer but this is only realised to a limited extent in the current workbench implementation. Likewise, EUDICO's Abstract Corpus Model provides an encapsulation layer between the user interface (client-side) and proprietary data representations (server-side). When looking at the reviewed tools there seems to be some tendency to use XML for data representation. XML is fine for data representation, but should never be something users have to bother about reading or writing as is the case for some of the reviewed tools.

(3, 4, 8) The tool may come with a number of best practice coding schemes for transcription and annotation. These should be described in a way which makes them comprehensible and easy to use. The MATE project has proposed a standard markup framework which facilitates a uniform and comprehensive description of coding schemes across annotation levels. The framework has been found useful for the description of spoken dialogue coding schemes at several different levels. However, it remains to be seen if the framework will also work for other NIMM areas.

(5, 7) Powerful query and information retrieval is needed for the user to exploit the data in the ways he wants. However, just as important as having powerful search mechanisms is the interface to the search tool. It must be easy to specify even fairly complex expressions and results must be presented to the user in a sensible, intuitive and easy-to-use way.

(7) In general, the interface to the tool should support the user as much as possible, be intuitive and be based on standard interface conventions which the user can be expected to know.

(6, 8) The tool should come with a set of predefined but flexible visualisations. In addition, it should be easy to define new views as necessary, e.g., to present annotations based on new coding schemes.

(9) Today, there is a wide-spread use of home-grown systems and formats. This makes it difficult to produce a standardised system which will still allow users to exploit the data resources they have built via other tools in whatever format they found appropriate. A solution to this problem may be tools for converting from the user's data format to the system's format. It should be made easy to write converters and add them since it is not plausible that the system will incorporate every conceivable converter.

As mentioned earlier, three of the reviewed tools/projects mention standardisation as a goal. Standardisation of annotation schemes and representation formats would facilitate the work of annotators and system developers. We believe standardisation requires a robust, flexible and powerful tool (toolset) which supports the standards aimed at. Given the current state of affairs, the first site or project which produces such a tool is likely to exert considerable influence on the general acceptance of standards. The need for annotation tools is clear and the users of annotation tools are out there in plenty. Today's users are aware that currently available tools are far from optimal, and they are eager for improvements. They are willing to try new tools, and are just waiting for something better than what they are offered today. It is up to the research and development community to try to meet their needs.

Acknowledgements

We gratefully acknowledge the support of the ISLE project by the European Commission's HLT Programme. We also grateful to Beppe Cappelli, John Garofolo, Leif Grönqvist, Christopher Laprun, Brian MacWhinney, Carol Neidle, Lucas Noldus, Khaldoun Shobaki, Hans Theuws, Eric Walter, and Peter Wittenburg who helped us by answering questions, giving advice on how to solve tool installation problems, and reading through and commenting on our reviews. Finally, we would also

like to thank Niels Ole Bernsen, Ulrich Heid and the ISLE NIMM Working Group partners in general for the input they provided to draft versions of this report.