Anvil 4.0

Annotation of Video and Spoken Language

User Manual

Michael Kipp

Graduate College for Cognitive Science
University of the Saarland
and
German Research Center for Artificial Intelligence (DFKI)
Germany
`kipp@dfki.de`

http://www.dfki.de/~kipp/anvil

April 9, 2003

Abstract

Anvil is a research tool for annotating digital video (QuickTime or AVI files) for research in areas like Linguistics, Human-Computer Interaction, Gesture Research or Film Studies. It provides a graphical user interface for creating annotation elements on temporal, hierarchical, completely user-defined layers. These elements are treated as complex objects that have typed attributes such as strings, booleans or token sets. Cross-level and within-level linkage can be realized through special attribute types. Non-temporal entities like relations, locations or objects can be annotated, too, and be linked with temporal elements. For managing multiple annotations Anvil offers the Project Tool that allows browsing and searching across annotations as well as exporting data to external statistics software like SPSS or Statistica. Anvil's platform-independence, generic approach, user-friendliness and flexibility makes it a highly effective tool for many research areas.

Contents

1 Introduction

1.1 Background

Anvil is a research tool for the analysis of digitized audiovisual data[1] (Kipp, 2001b). It allows you to transcribe human behavior and other visually accessible information in temporal alignment with speech and other auditory signals. It thus frees researchers in Anthropology, Gesture Studies, Psychology etc. from the need to work with video recorders and note-pads, taking down timestamps or frame numbers with pencil and paper.

The first aim of Anvil is to ease the transcription process itself by making the encoding of elements (*behavioral units*) as intuitive and fast as possible. The range of these elements, the actual *annotation scheme*, can be freely defined by the user. Second, Anvil was to provide the most informative *view* on the ongoing transcription. In our case, a transcription is an encoding of temporally parallel information, like speech and gesture, on multiple layers. This has often been compared to the way instruments are instructed in parallel in a musical score ("Partitur" in German). The most informative view on such a transcription looks very much like a musical score: layers are listed one below the other, running from left to right. Elements are depicted as boxes (rectangles) whose left and right borders correspond to their start and end points on a common time axis, the width thus being the duration of the element. Adding labels and colors to these bars allows intuitive comprehension of such a behavioral "Partitur" where temporal relationships, categories and durations can be captured at a glance.

On a technical side, Anvil relies on two word-wide standards: The *Java* programming language and the XML markup language. Thus, Anvil is completely written in Java and the video processing package JMF (Java Media Framework). Therefore, Anvil can run on any platform, be it Windows, Linux, Solaris or Macintosh[2]. Anvil's files are all based on XML which means that Anvil users can exploit the many tools that exist for the manipulation and transformation of XML files. **Did you know that almost every Internet Browser (Netscape, Internet Explorer) can read and display XML files? Try it with your Anvil annotation and specification files!**

Anvil does **not** provide convenient facilities to transcribe speech or other phonetic data like intonation. It is also not planned to include such capabilities because there are very good tools around for this (PRAAT, XWaves) and sound/video transcription do not really mix very well in a single tool because temporal granularity is much finer in speech transcription (22,000 frames per second) than in video transcription (25 frames per second). Anvil does also

[1]Common video formats like AVI and QuickTime are supported. See the Anvil website under `http://www.dfki.de/~kipp/anvil` for more information on formats and codecs.

[2]Anvil has been seen running on a Macintosh but I do not regularly check how easy it is to install there. For all other platforms (Windows, Linux, Solaris) installing Anvil is very easy.

not provide any kind of statistical analysis of the annotated data. This has to be done using other software like SPSS or Statistica (both commercial products).

Anvil was written as part of a PhD project on nonverbal communication at the University of the Saarland (Kipp, 2001c, and Kipp, 2001a) on a grant by the DFG (Deutsche Forschungsgemeinschaft) and builds on experiences with mass corpus annotation of dialogue acts within the speech-to-speech machine translation project VERBMOBIL (Alexandersson et al., 2000). It has been actively used to encode video samples of a German TV show with nonverbal behaviour and linguistic information.

The system was developed on a Pentium III (500 MHz) with 256 MB RAM, and successfully tested on Windows (98, NT, 2000, XP) and LINUX platforms. Since it is written in Java2 with JMF 2.1.1 (Java Media Framework) it should run on any platform that has those components installed.

This document is meant to serve as an installation guide, user manual and technical specification in one. Since I am aware of the fact that potential users of Anvil come from a wide range of fields, some more, some less familiar with certain concepts, I included notes in italics which are not of urgent general interest but may prove to useful to the more technical minded readers. If anything remains unclear, please send questions, remarks and suggestions to `kipp@dfki.de`

1.2 How to Use This Manual

This manual tries to be as complete as possible and may, in its current shape and size, deter you from reading anything at all. So let me give some hints how to read this manual for exploring Anvil most efficiently. For users familiar with Anvil I recommend looking at Appendix A to find out what features are new to Anvil 4.0.

If you have never worked with Anvil before you may want to try this order:

1. Read the next Section 1.3 "Installation" for the installation of Anvil.
2. Download the sample files on the Anvil download page and open them in Anvil to get used to navigation and manipulation of annotations. Look at Section 4 "User Interface" if you have trouble understanding the interface.
3. Proceed with Section 5 "Working in Anvil" where you will learn how to create an annotation with the predefined schemes.
4. Look at Section 3 "Concepts" to learn about the principal structure of annotations.
5. Read Section 8 "Writing a Specification File" carefully to implement your own annotation scheme. It is a good idea to take the predefined

samples in the "spec" directory as a starting point and to manipulate those step by step. Remember to give a new file name to each new specification that you create so that you can always get back to older versions.

6. Once you have coded a number of files you may want to consider the project functions outlined in Section 7 "Working with Projects".

1.3 Installation

This section explains how to install Anvil on Windows (95, 98, NT, 2000, XP) machines[3]. Before installing Anvil however, there are two other software systems that Anvil needs: Java2 and JMF (Java Media Framework). Both systems are public domain, i.e. they are free and can be downloaded under the addresses given on the Anvil download page. To install them, copy the respective files on your hard disk and execute them. An installation dialogue will guide you through the process. When asked to restart your computer, please do so. Otherwise, there could be problems with Anvil's installation.

Windows 95/98/NT users will also need the `WinZip` software which is already installed on most machines (have a look in your "start" menu). If not, you can download it from `http://www.winzip.com`. In Windows XP this software is in-built.

> **Java2** *is a platform-independent programming language. Anvil is completely written in Java and therefore runs on any machine supporting Java. The "2" in Java2 refers to the latest language specification of Java. To run Java programs you need to install a Java Virtual Machine, such as the one offered (for free) by Sun's Java Development Kit (JDK) version 1.4 or higher. This JDK 1.4 software is what you need to download and install before installing Anvil.*
>
> **JMF** *(Java Media Framework) is an extension for the Java language provided by Sun. It enables the use of video and sound facilities in Java. JMF is offered in a stable version for Windows and Solaris; for Linux there is a beta version provided by Blackdown (all free). It is yet unclear how stable JMF (in the "all-Java" version) runs on Macintoshes. For Anvil, I recommend to install JMF version 1.2.2 or higher.*

Having successfully installed Java2 and JMF, you can proceed with installing Anvil. First, create a directory, henceforth called "Anvil directory". This could be, on a Windows computer:

```
C:\program files\Anvil 4.0
```

Copy the file `anvil40-package.zip` to this directory.

[3]Unix (Solaris, Linux) users can have a look at the `README` file. Anvil *does* run under Linux and should run under Solaris. Whether it works on a Mac is unsure.

Second, start WinZip and open `anvil40-package.zip` in it. Extract all files to the Anvil directory. Windows XP users simply click on the `anvil40-package` file and Windows will show the contents. Copy all the files/directories that you see into the Anvil directory.

Finally, execute `install.bat`, located in the Anvil directory. The response will be a window that asks you to input the amount of working memory (RAM) of your computer (e.g. "512MB" or "1.4GB"). Anvil then informs you of successful installation. You will now find the new file `anvil.bat` in the Anvil directory. By executing it you will start Anvil.

The file `install.bat` can be called again if you wish to adapt the working memory. If ever you run into memory problems (an "out of memory" problem signalled by Anvil), then come back to `install.bat` and try a bigger number, even exceeding your true working memory.

2 Making Video Files

Since many people work with analogue video (e.g. VHS) to start with, I will give some guidelines how to get such videos into a digital form that can be used with Anvil. The process of digitizing video is also called "capturing".

2.1 Equipment

With the following kind of equipment you will be able to digitize analogue videos:

- Reasonably fast PC (800 MHz or more)
- TV card (e.g. by Pinnacle or WinTV, costs around $100)
- Video editing software like "VirtualDub"[4] (free!), "Premiere" (Adobe) or "Final Cut Pro"
- VCR (normal video tape recorder/player)

Macintosh users can visit `http://talkbank.org/digitalvideo` for detailed information on video digitization with a Mac. Mac users can work with a "hardware converter box" instead of a TV card (e.g. Sony sells them for around $500) in connection with the firewire interface.

[4]http://www.virtualdub.org

2.2 Processing

I highly recommend working out one first “perfect” digitization with a very small sample that turns out to be suitable for Anvil (see step 5) before starting to digitize your whole video collection. Take notes of how you set the options of the video software (see step 3) for later reference. You may need some experimentation there.

1. Plug the output of your VCR to your computer’s TV card. Best to use a SCART (VCR) to “composite” cable. The “composite” side has three chinch plugs (red and black for audio, yellow for video).

2. Start your video software and go to ”capture” mode (in Premiere this is `File > Capture > Movie Capture`).

3. Make some adjustments (try clicking the right mouse button while over the capture screen to get a menu) regarding the format you want to digitize your video to. For Anvil I recommend the following:

   - Screen Size: I recommend 384x288 (European TV half size) or 320x240 (American TV half size) – double values (768x576 or 640x480) if you really need the detail but be aware that you need a very powerful computer to cope with the increased data processing, and also, a large video pane leaves little room for Anvil’s Annotation Board (see p. 14).
   - Capture Format: set it to Video for Windows (AVI) or QuickTime (MOV)
   - Codec: Cinepak or Intel Indeo 5.0 (for other supported codecs follow the link on the Anvil homepage/download page)

   Leave other options (sound etc.) as they are. You can play around with things like ”video compression rate”. Such adjustments depend on the TV card, software, video format, and codec you use.

4. Insert your analogue video tape into the VCR and play it. You will see it in your software, too. Then, you can press the “Record” button anytime you like to start digitizing the portion you want. Note that your computer usually does not react immediately to your signal so press it somewhat in advance. Stop recording with the Escape key or pressing a mouse button, depending on your software.

5. Save the recorded video to a file (in Premiere: `File > Save`) and check the size of the file. It can range from 15 to 30 MB per minute of recorded video. If the size is much larger, play around with recording options (rate of compression) to arrive at a satisfying result. Also try loading the file to Anvil. It it does not work then Anvil does not support the codec. Try again with a different one.

3 Concepts

I will start out by presenting Anvil's underlying framework for annotations 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.

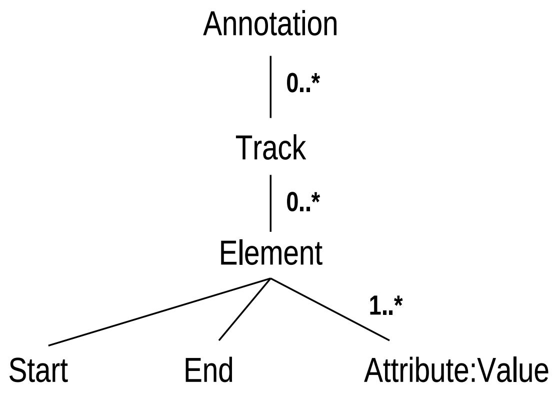


Figure 1: Annotation object model.

Anvil's overall design is object-oriented, the largest unit being an *annotation*. This annotation object holds a number of *tracks*. Each track represents a type of entity you want to encode (words, eye blinks, gestures) and contains the entities themselves, the so-called *elements*. Their contents can be specified by an arbitrary number of attribute-value pairs. Figure 1 shows the abstract object model[5].

The annotation depends on a *specification file* which contains a formal description of tracks, attributes, possible values etc.

3.1 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 that can 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 user-defined tokens. For instance, the attribute "emphasis" has a set of possible labels: `strong`, `moderate` and `reduced`. Other value types are: String, Boolean, Number, MultiLink and ReciprocalLink (see Sections 8 and 5.3 for details). On top of attribute-value pairs, an arbitrary number of comment lines can be added to each track element for spontaneous, free-form remarks.

[5]This figure ignores *groups* for simplicity.

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 analyzed. This type of track is called a *primary track*. The other two types are secondary types because its 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.

Singleton tracks can are in a way equivalent to attributes of their reference track. In the above example where part-of-speech labels are encoded in a singleton track, the same could be done in an attribute "part-of-speech" of the transliteration track. The difference lies in the higher visibility of a singleton track and in different ways to access elements when searching.

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).

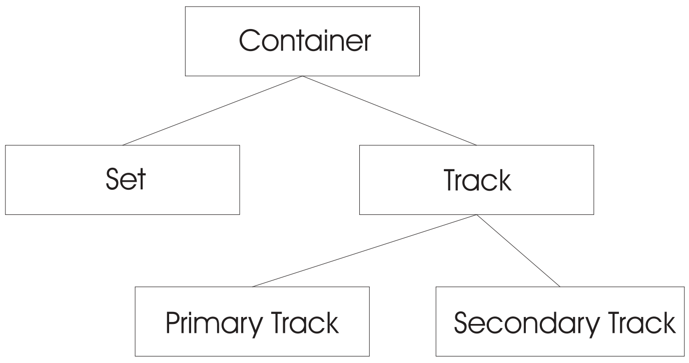


Figure 2: Containers, Tracks and Sets.

3.2 Groups

Tracks can be grouped together by a *group node* (something like a directory folder) which is equivalent to a track but cannot hold any elements and is therefore just a structural entity. You can use this for didactic or ergonomic reasons, e.g. to put together all linguistic tracks (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.

3.3 Sets

Tracks contain temporal elements that all have a start and end time. Even elements of secondary tracks have a beginning and an end, only it is determined by the referred to elements. For some purposes it makes sense to encode *non-temporal entities*. For instance, if a teacher points at different objects that lie on a table (say, a book, a journal and a CD), you may want to link up his pointing gestures with representations of these objects. For these representations you do not want temporal information. Instead, you want to handle them, e.g., in a list. On the other hand, you can represent these things with the same kind of "elements" that are contained in track, only without the time information. The solution is the concept of the "Set" which mirrors that of the track, only that the time information is missing from its set elements (Martin and Kipp, 2002).

The more abstract concept above tracks and sets I call the "container" (see Figure 2). Tracks contain temporal elements, sets contain non-temporal elements.

3.4 File Organization

File organization (Figure 3) is centered around the Anvil data 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 is contained in the "head" section of the annotation file. Data like coder name, date of encoding etc. is kept in the annotation file as well.*

An Anvil project keeps a list of files as indicated in the figure. Project files are maintained by the Project Tool (Section 7, p. 29).

Other relevant files are the HTML files created by Anvil's coding manual generator (see Section 5.11) and those files that can be imported into the current annotation: Text grid files produced by PRAAT will be fused into a transliteration track, rs2 files produced by the RSTtool are inserted in the RST track (see Section 6).

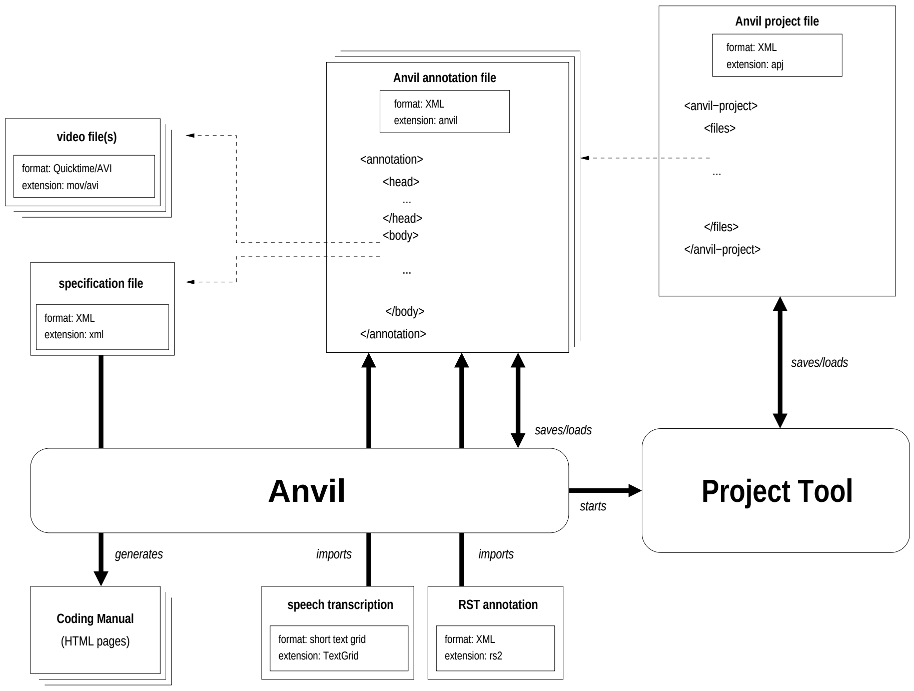


Figure 3: File organization

4 User Interface

Let me give you a quick tour of Anvil's graphical user interface that you can see in Figure 4. It has five components: the Main Window, the Video Window, the Element Window[6] and the Annotation Board.

4.1 Main Window

The Main Window is the first window that you will see after starting Anvil. It holds the main menu bar and, underneath, a *tool bar* with icons that provide short-cuts to Anvil's most vital functions. A text display lists user actions and other information (e.g. the exact format of the video file).

Near the bottom the Main Window displays the specification file of the currently loaded annotation. When the annotation is modified and not yet saved you will find the word "modified!" in red.

Located at the very bottom are the video controls, including buttons for frame-by-frame stepping. You also find the current time and frame of the video above the controls.

[6]Known as "Track Window" in Anvil version 3.5 and older.

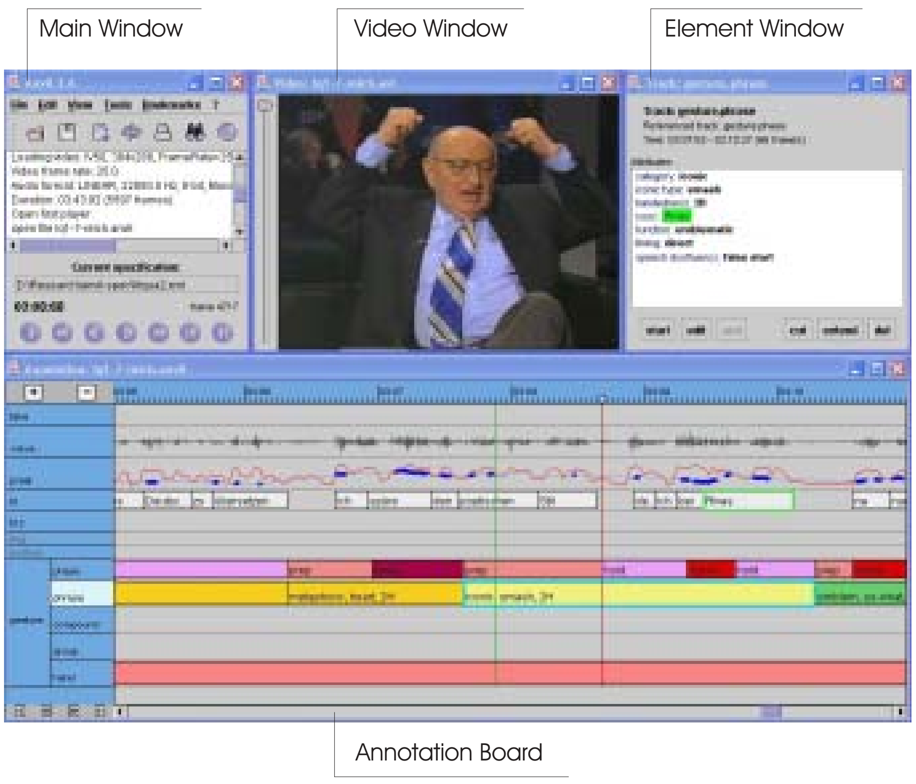


Figure 4: All windows of the Anvil system.

4.2 Annotation Board

The Annotation Board (Figure 5) is really the most important window since all the coding takes place here. It gives you a time aligned view on all tracks and their 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 8). 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 (small ticks represent *video frames*). 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 *playback 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 playback line to be positioned on the click point and the respective track to be selected. For better viewing, whole groups can be hidden ("collapsed") by deselecting the group in the **view** menu (Main window).

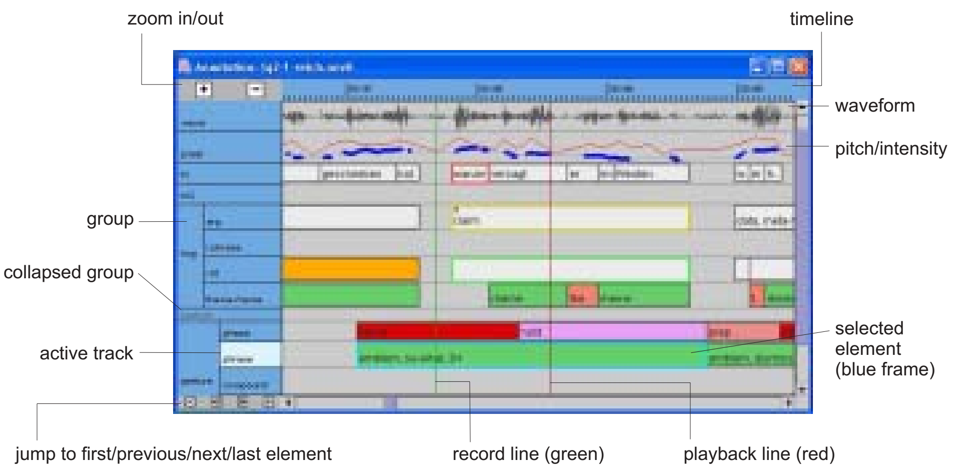


Figure 5: Annotation Board

The track elements are displayed with the attribute values as specified by the user (see Section 8.5). For instance, in the transliteration track (abbreviated "trl") the attribute "word" is displayed. One attribute can also be displayed by means of color-coding. Each value can be assigned a specific color that is used to fill the track elements' box (also Section 8.5).

4.3 Element Window

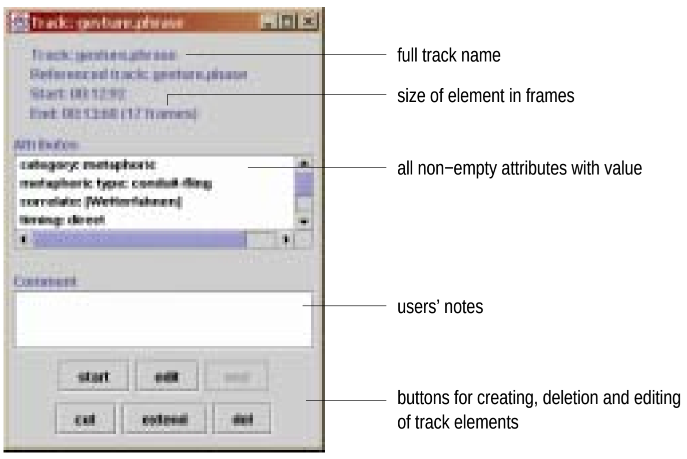

full track name
size of element in frames
all non−empty attributes with value
users' notes
buttons for creating, deletion and editing of track elements
start
edit
cut
extend
del

Figure 6: Element Window

The Element Window (Figure 6) gives you some information on the active container (track or set) and its currently selected element. In a track, the *currently selected element* is the element in the highlighted track which is intersected by the playback 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 active track (e.g. “gesture.phrase”), 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 playback line at the desired end time) which will trigger the element edit window (see Section 4.4 below). Other functions are the deletion and editing of existing elements and the linking of subsequent elements to element groups.

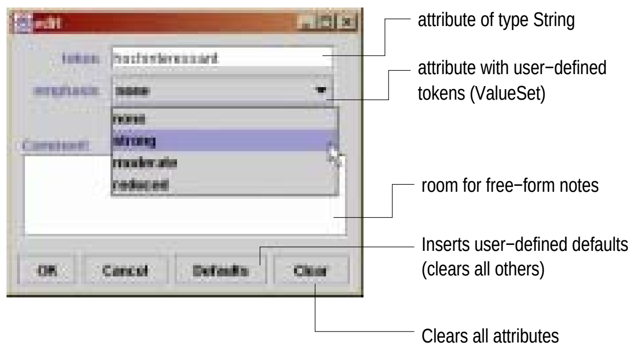


Figure 7: Example edit window

4.4 Element Edit Window

The edit window pops up when the user wants to create a new or edit an existing track element (see Figures 7 and 8). It shows all the attributes applicable for the current track. The value for an attribute is input depending on the attribute value type (see also Section 8.2). Table 1 shows the different value types and their GUI input look. If you have included any documentation in your specification file you can look at it online by pressing the info buttons (see Figure 8). Anvil will show you your documentation in table form like in Figure 9.

Comments concerning this element can be inserted into the lower text area. Elements with non-empty comment will be marked with a little square on the Annotation Board.

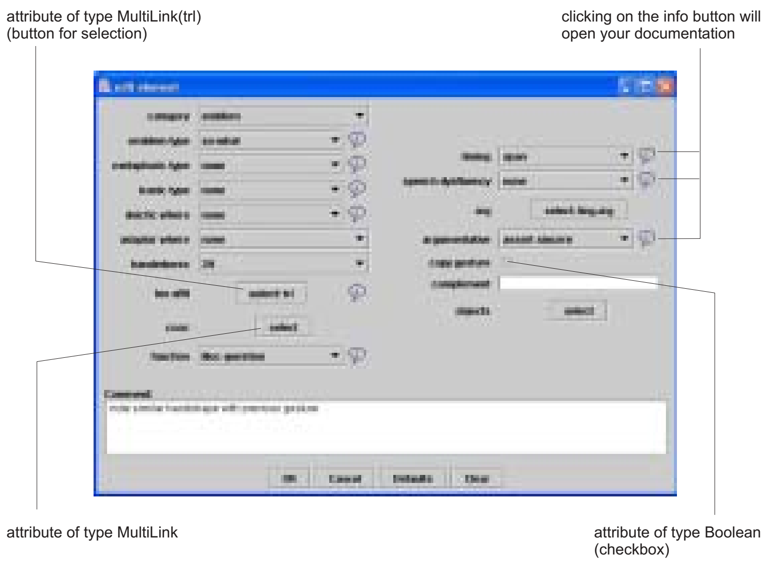
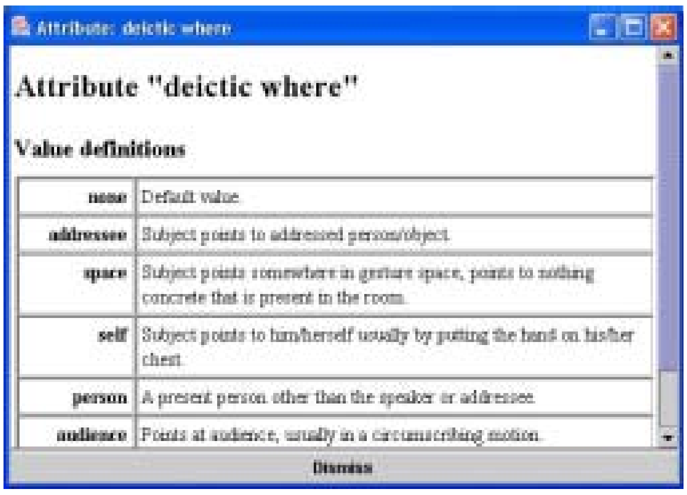

attribute of type MultiLink(trl)
(button for selection)
clicking on the info button will
open your documentation
attribute of type MultiLink
attribute of type Boolean
(checkbox)

Figure 8: Example edit window



Attribute "deictic where"

Value definitions

none	Default value.
addressee	Subject points to addressed person/object.
space	Subject points somewhere in gesture space, points to nothing concrete that is present in the room.
self	Subject points to him/herself usually by putting the hand on his/her chest.
person	A present person other than the speaker or addressee.
audience	Points at audience, usually in a circumscribing motion.

Figure 9: Example attribute documentation window

5 Working in Anvil

This section gives you concise "how to" instructions for most of Anvil's functions. Read the following "preliminaries" to become familiar with some Anvil

value type	*GUI input*
String	string input field
Boolean	check box
MultiLink	selection window plus Annotation Board
ReciprocalLink	selection window plus Annotation Board
ValueSet	roll-down option menu
Number	number slider

Table 1: Attribute value types

specific notions and then. . . have fun!

> **SAVETY NOTICE:** When working with Anvil, please remember to back-up your valuable annotations from time to time. Simply copy all files with extension "anvil" to another directory or a disk. Although so far, I haven't heard of any serious data losses with Anvil, it is always possible that, for instance, your system crashes while Anvil is writing out data which almost certainly will make the file unusable and so possibly destroy days or weeks of hard work.

5.1 Preliminaries

Take a look at Figure 4 to get an impression of the overall system and the names of the different windows: Main Window, Annotation Board, Video Window, and Element Window.

For editing, two notions will be referred to in this document: The *active track* is highlighted in light blue in the left part of the Annotation Board. You activate a track by clicking on the track's name. The *selected element* is the box framed in light blue on the right hand side of the Annotation Board. You select an element by clicking on it. Selecting an element makes Anvil also activate the respective track.

5.2 Creating Annotations

Before you can create a new annotation, Anvil needs to know two things: what video you want to code and which annotation scheme, called *specification*, you like to follow. So first open a video file using **File>Open** in Anvil's menu bar. Anvil will ask you for a specification (to change the offered one, press "browse") and other, optional information (your name, a comment on this annotation).

If you close the current annotation and want to start a new one on that same video, select **File>New annotation**. You can create new annotations at any time provided that there is a video file opened.

Writing specification files is most easily learned by looking at existing ones. There are a number of sample specification files in the subdirectory "spec" of the Anvil directory. To create your own specification, read at Section 8.

5.3 Adding Track Elements

Adding track elements is your main concern when coding. It all happens on the Annotation Board. The control buttons for adding can be found in the Track window (top right) or in a context menu that pops up when you press the right mouse button while being on the Annotation Board.

Primary Track In primary tracks, elements are time-anchored. In order to insert a new track element you mark begin and end time by placing the green *record line* at the beginning, the red *playback line* at the desired end point. The red playback line always follows the video's frame position in playback (it can also be dragged with the mouse, the video will follow). 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 context menu (right mouse button) or in the Element Window. You then place the red line at the desired end point and press **end**. An edit window will appear where you can specify the contents of your new element (see Section 4.4).

Singleton Track In a singleton track, each element corresponds to one and only one element in the respective reference track. To create a new singleton element, place the playback line on the respective element in the reference track but be sure that the singleton track is active. Press **tag** in the Element Window or in the context menu and the edit window will appear.

Span Track In a span track, each element corresponds to a sequence of elements in the respective reference track.

You insert a span element by specifying the first and the last element of the reference track. You do this, again, by placing a *record line* upon the first element (no matter where exactly as long as the line runs through the element). Then, position the *playback line* over the last element and select **end** (again, be sure that the span track is active). An edit window will appear to ask for the attributes' values.

MultiLink and ReciprocalLink Attributes Inserting attribute values is trivial for Strings, Booleans and ValueSets. It is different for links (MultiLink and ReciprocalLink). To specify the links, press the **select** button located right of the attribute name (Figure 8, p. 17). Two things will happen: (1) a *select* window appears that keeps track of the selected elements (Figure 10, p. 20), and (2) the Annotation Board will be free for navigation. Clicking

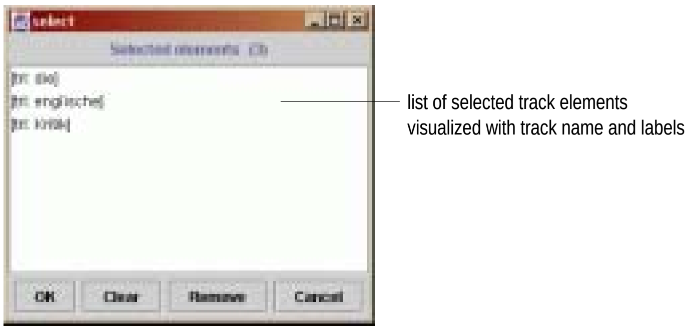


Figure 10: MultiLink select window

on elements will add them to the select window (or remove them if they are already in). Selecting OK in the select window transfers the selection to the currently edited element.

Note that a MultiLink/ReciprocalLink attribute takes an **ordered set** of links. "Ordered" means that the order of selection is reflected in the order of links. "Set" means that every link can only occur once. Adding the same link twice is technically not possible (because clicking a highlighted element de-selects it) but can happen when working with ReciprocalLink types. Even then, adding the same link twice does not change the set (not even the order).

Adding reciprocal links[7] to element A makes Anvil add respective links to the linked up elements. Imagine, for example, that you add the elements $L_1, \ldots, L_n$ to attribute "foo" of A. Attribute "foo" is of value type `ReciprocalLink(baa)`. In this case, Anvil will look for an attribute "baa" in each element $L_1, \ldots, L_n$ and insert A each time it finds such an attribute (and do nothing in case there is none).

This way, you can realize *symmetrical* attributes like "synonym" of type `ReciprocalLink(synonym)`, or *asymmetrical* attributes like "hyponym" of type `ReciprocalLink(hypernym)`.

5.4 Manipulating Track Elements

Existing elements can be changed in three ways. Contents can be changed with the **edit** option available in Element Window and context menu, and is straightforward to use. To change the position of begin and end times, you can **cut** or **extend** elements.

Cutting elements This option lets you cut off part of an element. For primary elements, place the red line where you want to cut the element and press **cut**. Anvil will ask you whether the red line is supposed to be the new

[7]Computer scientists can thinks of reciprocal links as *backlinks*

start or the new end location. It will then reduce the size of the element accordingly.

For span elements, which consist of a number of elements $R_1, \ldots, R_n$ on the reference track, place the red line on some element *between* R_1 and R_n (while making sure that the span track remains active) and press **cut**. Anvil will ask you whether the marked place should be new start or end, resizing the element accordingly.

Singleton elements can obviously not be cut.

Extend elements While **cut** allows you to reduce the size of existing elements, **extend** serves to enlarge elements. Place the red line left or right of the element you want to enlarge, exactly where you want the new start/end border to be (this can even be within another element!). Anvil will ask you whether you want to extend the previous element (left of red line) or the next element (right of red line) and then adapt the size. If you placed the red line on a bordering element, this element will be *cut* or even deleted if you cut away almost all of the element (in case of deletion, Anvil will ask you to confirm the change).

Note that in the current version, Anvil might behave unexpectedly if you place the red line on *borders* of elements and press **extend** (which does not make much sense anyway). It will, however, not produce any damage to your annotation.

Singleton elements can obviously not be extended.

5.5 Playing a Segment

If you want to playback a certain segment of the video again and again, you can place the record line (green) at the beginning of the desired segment, the play line (red) at the end, and select **play line-to-line** in the context menu (right mouse button on Annotation Board). Anvil will playback the portion between green and red line, stopping as soon as the red line is met.

If you want to playback an annotated element in a track, simply select the element and choose **play element** in the context menu.

5.6 Navigation and Short-cuts

Finding Elements You can search for specific track elements in the currently active track by selecting **Edit>Search current track** in the main menu bar. Anvil will ask you for search criteria and provide a hit list. The hit list can be used to directly jump to the found elements. Since all this works like an analogous Project Tool function it is described in Section 7.3.

Element Hopping Within the active track you can jump to the first or last element by clicking on the respective symbols on the Annotation

Board (lower, left corner, see Figure 5). Moreover, you can jump to the next/previous element with respect to the currently selected track element.

Keyboard/Mouse Shortcuts Table 2 shows the key/mouse commands valid on the Annotation Board.

key or mouse command	*action*
`2 x Mouse-Left`	position green record line (like "start")
s	position green record line (like "start")
`1 x Mouse-Middle`	create track element (like "end")
e	create track element (like "end")
del	delete selected track element
enter	edit selected track element
space	edit selected track element
home	go to first track element
end	go to last track element
Ctrl + ↑	move active track one up
Ctrl + ↓	move active track one down
Ctrl + ←	jump to previous track element
Ctrl + →	jump to next track element

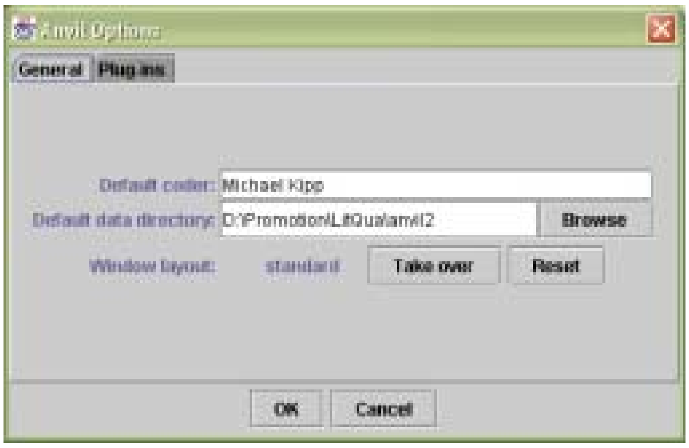
Table 2: Annotation board shortcuts

Bookmarking Bookmarks can be used to mark locations of particular interest in a video. They work very much like bookmarks in an Internet Browser. You access them through the main menu bar. Place the *playback line* at the desired location and select **Bookmark>Add bookmark**. A dialogue box will ask for name and description, and on clicking OK a small blue triangle will show up in the time bar. With **Bookmark>Edit bookmarks** you can remove/rename existing bookmarks.

Bookmarks are stored together with the encoded data in the Anvil file.

5.7 Customizing Anvil

You can tell Anvil to remember the position of your windows and change other defaults. You find the respective console in **File > Options...** (see Figure 11). To store the position and size of your windows (Main, Video, Element Windows and Annotation Board) you click on **Take over**. If you want to switch back to Anvil's standard layout click **Reset**.



Default coder: Michael Kipp
Default data directory: D:\Promotion\LitQual\anvil2
Window layout: standard

Figure 11: Anvil's options adjustment window: **File > Options...**

5.8 Registering Plug-ins

To register a plug-in, start the Options Window with **File > Options...** and click on the **Plug-Ins** tab (see Figure 11). You enter a new plug-in by clicking **add** and inserting name and classpath of the plug-in. The plug-in will appear in the **Tools** menu when you start Anvil the next time. Note that you have to make sure that your plug-in's class file(s) have to be located in the "plugins" directory. Alternatively, you can adjust the classpath in the Anvil startup script to point to your class file(s).

5.9 Printing Annotations

The printing facility does not always work (depending on operating system or hardware?). Unfortunately, I have not been able to spend any time to check on this, so if you cannot print you will have to resort to screen shots.
The current version offers a first version for printing the annotation. When having loaded an annotation, choose **File>Print annotation board** in the Main Window menu bar. Anvil will ask for printer and paging info. Due to a bug the maximal page will appear as "9999" (under Windows2000). But, of course, Anvil will only print out as many pages as necessary for your current annotation. I highly recommend, though, to first only print one page (specify page "1" to "1") to get an impression of how the print-out looks in size. The Main Window trace text (upper left window) will tell you how much pages Anvil would print in total.

The print-out looks like the Annotation Board without playback/record lines and without highlighting. It comes in landscape orientation and multiple pages can be glued together to get a seamless panoramic view of the whole work.

5.10 Exporting Text

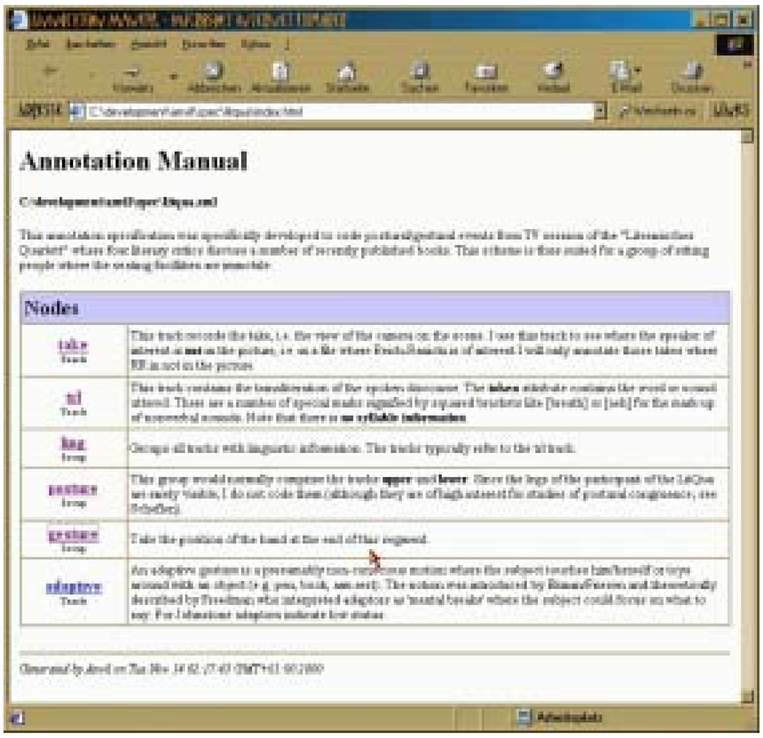
Once you have imported speech transcription as outlined in Section 6, you may want to export the transcription as a simple ASCII text file. You can do this by selecting **File>Export: track to txt** in the main menu. Anvil will ask you which track to export. The selected track (for example, the one containing your speech transcription) will be written to the chosen file. In that file, you will find all elements in correct order, separated by blanks. Each element is represented the same way as in the respective boxes on the Annotation Board (see Section 4.2).

Exporting the text of the speech transcription is, e.g., important for RST encoding (see Section 6). way as it is displayed on the Annotation Board (see Section 4.2).

The exported text is not meant for statistical analysis. How to export data for statistical analysis is described in Section 7.2, p. 30.

5.11 Generating a Coding Manual

For systematic annotation is it essential that the coding scheme's 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>...</doc> tags, see Section 8.7) you can have an HTML manual generated which has the look-and-feel of javadoc (see Figure 12). The advantage of HTML being that you can browse the electronic manual on screen while in the process of annotation. Selecting **Edit>Specification>Create manual** will make Anvil produce an HTML manual of the current default specification. If the current default specification is called "foo.xml", located in directory "specdir", then you can find the HTML manual in subdirectory "specdir/foo".



Annotation Manual

C:\development\anvil\spec\litqua.xml

This annotation specification was specifically developed to code postural/gestural events from TV session of the "Literarisches Quartett" where four literary critics discuss a number of recently published books. This scheme is thus suited for a group of sitting people where the seating facilities are immobile.

Nodes	
take Track	This track records the take, i.e. the view of the camera on the scene. I use this track to see where the speaker of interest is **not** in the picture, i.e. on a file where Reich-Ranicki is of interest I will only annotate those takes where RR is not in the picture.
wl Track	This track contains the transliteration of the spoken discourse. The **token** attribute contains the word or sound uttered. There are a number of special marks signified by squared brackets like [breath] or [ueh] for the mark up of nonverbal sounds. Note that there is **no syllable information**.
ling Group	Groups all tracks with linguistic information. The tracks typically refer to the wl track.
posture Group	This group would normally comprise the tracks **upper** and **lower**. Since the legs of the participants of the LitQua are rarely visible, I do not code them (although they are of high interest for studies of postural congruence, see Scheflen).
gesture Group	Take the position of the hand at the end of this segment.
adaptive Track	An adaptive gesture is a presumably non-conscious motion where the subject touches him/herself or toys around with an object (e.g. pen, book, armrest). The notion was introduced by Ekman/Friesen and theoretically described by Freedman who interpreted adaptors as 'mental breaks' where the subject could focus on what to say. For [illegible] adaptors indicate low status.

Generated by Anvil on Tue Nov 14 02:17:43 GMT+01:00 2000

Figure 12: Start page of automatically generated coding manual.

6 Importing Data

Anvil can import annotations (for example speech transcription), pitch and intensity data from PRAAT, and speech transcriptions from XWaves.

6.1 PRAAT Transcription

Anvil supports importing data from PRAAT[8]. More precisely: Anvil can import arbitrary *interval tiers* from a PRAAT *short text file*. This section will explain step by step how to do this for the case of speech transcription. Similar procedures apply for any other kind of data that you code in PRAAT (syllables, intonation labels etc.).

I start out by describing very briefly how to annotate speech signals in PRAAT. First of all, extract the sound track from your video file, saving it

[8]PRAAT, developed by Paul Boersma and David Weenik, runs on many platforms (Windows, Unix, Mac) and is freely available. Contact Paul Boersma `paul.boersma@hum.uva.nl` and visit `http://www.fon.hum.uva.nl/praat` for more information.

in, for instance, Microsoft's "wav" format (or any other format PRAAT can read).

In PRAAT, you have a main window which displays a list of PRAAT *objects*. Any sound file that you load, any annotation that you create, will become a PRAAT object. Now proceed as follows:

1. Select **Read>file** in the menu bar and choose the sound file. It will appear as a sound object in the list.
2. Click on **label & segment**, selecting **To Text grid**. In the dialogue box, erase the default values and insert "words" in the **Tier names** input field. Leave the **Point tiers** field empty.
3. The last step produced a second entry in the object list, a Textgrid object. Holding down the CTRL-key, select both objects in the list.
4. Click on **Edit**. The annotation window with waveform (amplitude over time) plus your "words" annotation tier are displayed.
5. Transcribe words. Consult the excellent document by van Lieshout (2000)[9] if you need help here.
6. Select the Textgrid object *only* in the list of the main window by clicking on it (**important!**[10])
7. Select **Write>write short text file** in the menu bar and choose a location in the ensuing file name dialogue.

In Anvil, you can import the PRAAT data to any *primary* track that has at least one attribute of String type. Select **File>Open** in Anvil and pick the PRAAT short text file (e.g. "foo.TextGrid"). You will be asked to choose a tier from the PRAAT file (in our example, there will be only the "words" tier) and the target track in Anvil. Then, you will be asked which attribute you want the words to go in. Having decided this, you will see your annotated words appear on the right track. Note that everything else in that track will be erased (though Anvil will ask your permission before).

6.2 PRAAT Intensity and Pitch

You can import intensity and pitch data that you computed in PRAAT into Anvil and let it be displayed as a track (Figure 5 on page 15). In PRAAT do the following:

1. Load sound file with **Read>Read from file...**

[9]Available under `http://www.fon.hum.uva.nl/praat`

[10]It is crucial that before the step there is only the Text grid object selected! Otherwise, nasty things will happen and your hours worth of annotation will irrevocably be lost.

2. Depending on the PRAAT version, do the following

   (a) PRAAT version 4.0 or newer:

      - Pitch: Click on "Periodicity" and select "To Pitch". Press "OK" in the upcoming dialogue window. The ensuing computation will take a while. When it is done, select the new Pitch object and click on "Down to PitchTier".
      - Intensity: Click on "To intensity" and press "OK" in the upcoming dialogue window. The ensuing computation will take a while. When it is done, select the new Intensity object and click on "To IntensityTier(peaks)".

   (b) PRAAT version older than 4.0: Analyze the sound object by clicking on "To Analysis..." (in newer PRAAT versions). Press "OK" in the upcoming dialogue window. The ensuing computation will take a while. Select the new Analysis object and click on "Extract pith tier"

3. Select the new PitchTier/IntensityTier object and save it with **Write>Write to short text file...**

In Anvil do the following:

1. Your specification must reserve a "speech analysis" track which is defined like this (in the example I call the track "praat" but that is up to you):

   ```
   <track-spec name="praat" type="speech analysis" />
   ```

2. Load your annotation with **File>Open**

3. Load the IntensityTier or PitchTier file with **File>Open** and click "OK" in the upcoming dialogue window. Anvil will display the respective contour in the reserved track.

The pitch contour will be displayed as shown in Figure 5 (page 15). Note that due to implementational reasons the recomputation while zooming in and out may take a while.

6.3 XWaves Speech Transcription

Import of transcription files from XWaves (Entropic) works similar to the PRAAT case (see above). Simply select the file, say "foo.words", with **File>Open** and select the Anvil target track and then, target attribute. Note that in XWaves one actually annotates time points which are translated by Anvil to intervals.

6.4 RSTtool Annotation

Anvil can also import Rhetorical Structure, a hierarchical markup of text organization (cf. Mann and Thompson (1988) for Rhetorical Structure Theory) that can be coded with the RSTtool[11]. Imported data is transferred to a secondary track in a flattened form (only rhetorical segments plus relation name and relation direction, i.e. forward/backward). There is a slight modification to RSTtool needed, though, since it does not produce 100% legal XML files. Contact me (`kipp@dfki.de`) for the modifications.

[11]RSTtool was developed by Mick O'Donnel and can be downloaded on `http://www.sil.org/linguistics/rst/micktool.htm` (version 2.5). It is written in Tcl/Tk 8.3 and runs virtually on any platform.

7 Working with Projects

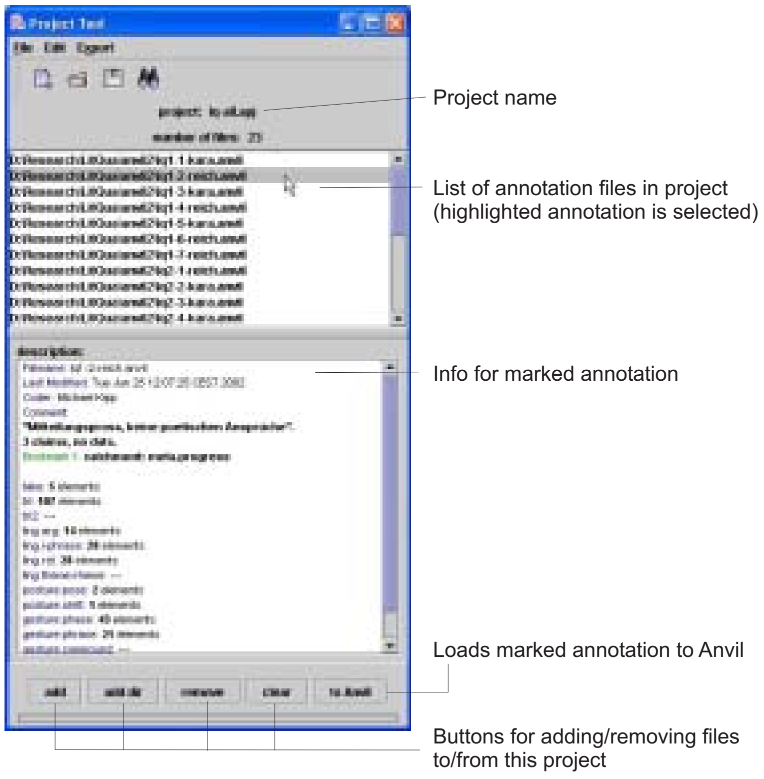

Project Tool
File Edit Export
Project name
List of annotation files in project (highlighted annotation is selected)
Info for marked annotation
Loads marked annotation to Anvil
add
add dir
remove
clear
to Anvil
Buttons for adding/removing files to/from this project

Figure 13: Anvil Project Tool.

Very often you want to look at more than one annotation. Say you have recorded and annotated a number of classroom sessions (teacher, students, blackboard) and are interested in looking at all the deictic gestures (pointing gestures) that occurred in these sessions. Or maybe you only want to look at those sessions with a particular teacher or those recorded in a particular semester. Whatever annotations you are interested in you can assemble to a list (of Anvil files) that we call a *project*.

The Project Tool (Figure 13) allows you to assemble such file lists or *projects* and store them for further usage. You start the Project Tool in Anvil, in the main menu under **Tools>Project Tool**.

Within a project you can search a specific track and locate the found elements in the annotation. Or you can export a track to a text table intended for statistical analysis.

Note It is necessary that all files in a project have the same specification file! This is logical when you want to search the same track in multiple files: equal specifications guarantee that this track exists in all the files. Anvil will complain if you try to add files with differing specifications.

7.1 Creating, Opening And Browsing a Project

To create a new project, simply keep adding files by pressing the `add` button at the bottom of the window, or add a whole directory of Anvil files with `add dir`. Remove undesired files with `remove` or all files with `clear`. Once you have all desired files in your project, save it with **File>Save** in the Project Tool's menu bar.

Next time you want to work with this project, use the **File>Open** option of the Project Tool's menu bar to load it.

Note that you can comfortably browse the files contained in the currently opened project. When you click on a file name, information about its tracks is displayed in the bottom text section. Also, you can load the currently highlighted file to Anvil by clicking the `to Anvil` button.

7.2 Exporting Tracks

If you need to analyze the elements of a specific track, say "gestures", across a number of annotations, you have to put the track data all the annotations to a single text file. This text file can then be analyzed by statistical software like SPSS or Statistica.

In the Project Tool's menu bar, select **Export>track table** and pick the desired track, the desired output format (SPSS or Statistica) and file name from the ensuing dialogues. Anvil will produce a text file containing a large table. In this table, all track elements are listed in rows. Each row contains (1) an identifier, (2) the element's start/end time, and (2) all attribute values. The first row contains the names of the columns, i.e. the names of the attributes. In SPSS and Statistica, the table can be imported, so that elaborate quantitative analysis can ensue.

For the SPSS output format you get numbers as attribute values. Anvil automatically converts ValueSet tokens into numbers. To know what numbers correspond to what tokens, Anvil produces an index file where, for all ValueSet attributes, a number to token mapping is printed. The file name of this second file is generated from the chosen file name: if you exported your table to, say, `foo.txt`, the index file will be called `foo_labels.txt`.

7.3 Finding Track Elements

With the Project Tool you can search for track elements across annotations. The resulting hit list can be used to navigate through annotations: simply click on a found element and Anvil will load the respective annotation and

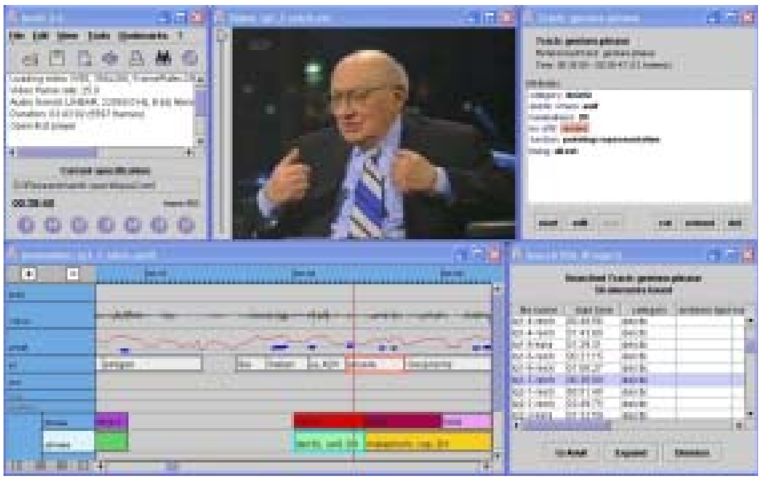


Figure 14: Anvil with integrated hitlist window.

position itself right on the element. This makes comparison of annotated elements fast and easy (see Figure 14).

In the Project Tool's menu bar, select **Search>Find track elements** and choose a track in the ensuing dialogue. The following Search window (Figure 15) allows you to look for elements sharing certain property predicates specified in the *goods* list $G = \{g_1, \dots, g_k\}$ and *nogoods* list $N = \{n_1, \dots, n_h\}$. A property predicate g_i has the structure $A = V$ where A is an attribute and V is a value. For predicates n_j this is a negation $A \neq V$. Anvil will display all elements with

$$(g_1 \vee \dots \vee g_k) \wedge (n_1 \vee \dots \vee n_h)$$

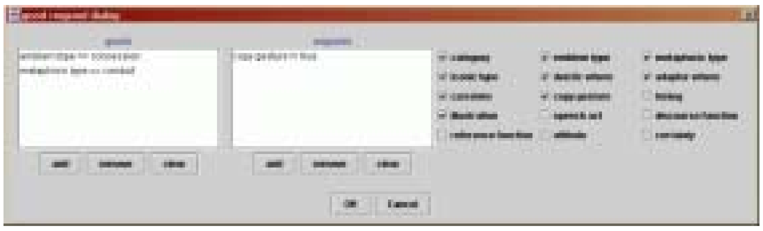


Figure 15: Search window.

You can change the way the elements are displayed with the display option checkboxes shown to the right of the Search window. The found elements (hit list) will be output as a table. Double clicking elements in this table makes

Anvil load the respective annotation and position itself on the right element. The same is done when clicking on the `to Anvil` button. The button `Shrink to fit` can be clicked to integrate the hit list window into the normal Anvil layout to make browsing more comfortable (see Figure 14).

8 Writing a Specification File

With the *specification file* you tell Anvil how your annotation scheme looks like, i.e. what tracks you need, how they are called, and what attributes and values their elements may have.

The language for doing this is XML (eXtensible Mark-up Language) which looks very much like HTML and should be easy to get used to, even if you do not know either XML or HTML. A sample file of a complete specification is printed in Appendix C (p. 49). If you have installed Anvil, you can find more examples of specification files in the subdirectory "spec" of the Anvil directory.

8.1 File Structure

Specification files have the standard extension "xml" like in "foo-spec.xml". When you open a file you will see that it has two main parts: the *head*, specifying the user's own value types, and the *body*, containing the track and group specifications. So each specification looks in principle like this:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<annotation-spec>
  <head>
    ...
  </head>
  <body>
    ...
  </body>
</annotation-spec>
```

The head part is optional and can be omitted. What this sample nicely demonstrates is the principle structure of XML files in general where pieces of code are bracketed by so-called *tags*. So the XML tag `annotation-spec` brackets the head part as well as the body part. The `head` and `body` tags bracket the sub parts of the specification. Bracketing works with an opening and a closing bracket, the latter being marked with a "/" (called "forward slash").

Let's start with the body because the function of the head will become clear only at a later stage.

8.2 Tracks and Groups

Specify a track with XML tag `track-spec`, including the name and the type (primary, singleton or span) as attributes. Note that **the track name must not contain a dot "."** because the dot is reserved as a special symbol.

If your track is of singleton or span type, specify its reference track in the `ref` attribute. Use the full name for nested tracks, e.g. "gesture.deictic" for track "deictic" in group "gesture" (see Section 3.2). Note that **secondary tracks must be defined after their associated reference track**. This is a convenient way of avoiding cycles (e.g. two secondary tracks depending on each other). The specification of the track's elements will be bracketed by the `track-spec` opening and closing tags:

```
<track-spec name="trl" type="primary">
   ...
</track-spec>
<track-spec name="rst" type="span" ref="trl">
   ...
</track-spec>
```

One or more tracks can be organized in a group by bracketing the track specifications in a `group` tag:

```
<group name="ling">
  <track-spec name="i-phrase" type="span" ref="trl">
     ...
  </track-spec>
  <track-spec name="rst" type="span" ref="trl">
     ...
  </track-spec>
</group>
```

The order in which you define the tracks and groups here will be the same as on the Annotation Board later on.

Edit option When adding elements in Anvil's graphical user interface (GUI), Anvil will automatically insert the values of the last added element as default values for the new one. If you want to switch this off, use the `edit-takeover` feature like this:

```
<track-spec name="trl" type="primary" edit-takeover="false">
   ...
</track-spec>
```

Track height Track are displayed in the GUI as bars with a certain height. You can customize this height by specifying a scaling factor, e.g. if you want a track to be twice as high as a normal track that factor would be "2" (or "1.5" for one and a half times as big). Specify this factor in the `height` feature. Anvil accepts height values between 0.7 and 2:

```
<track-spec name="trl" type="primary" height="1.2">
   ...
</track-spec>
```

Waveform track To display the waveform of a video's audio data, you create a track of the special type "waveform". You have to give this track a name, you can position it anywhere you like, you can scale its height, just as with any other track:

```
<track-spec name="audio" type="waveform" height="1.5" />
```

Note that Anvil will need more time to load a video when this feature is switched on. So if you are bothered by slow loading times, remove the waveform track.

Collapse by default If you want to have certain groups *collapsed* (Section 4.2) by default you can tell Anvil so:

```
<group name="ling" collapse="true">
  ...
</group>
```

This makes Anvil collapse group `ling` each time a new Anvil data file is loaded.

8.3 Sets

A set is a container for non-temporal objects (see Section 3.3). You define a set called, for example, "objects" like this:

```
<set-spec name="objects">
   ...
</set-spec>
```

A set contains attributes just like a track. But unlike tracks, a set cannot be grouped with the XMLgroup tag.

8.4 Attributes

Within the `track-spec` opening and closing tag you can define attributes and corresponding *value types*. There are five value types: String, Boolean, Number, MultiLink and ReciprocalLink. When using these types, the attribute tag is usually an empty tag. For `String` and `Boolean` types simply write:

```
<attribute name="token" valuetype="String" />
<attribute name="repetition" valuetype="Boolean" />
```

For `Number` types you specify an interval. Here we define the interval between 0 and 4 (including borders):

```
<attribute name="token" valuetype="Number(0,4)" />
```

A `ReciprocalLink` is a link where a reciprocal relationship between links is defined. For instance, "hyponym" is reciprocal to "hypernym", which is an example for an asymmetrical relation:

```
<attribute name="hyponym" valuetype="ReciprocalLink(hypernym)" />
```

The attribute "synonym" is an example for a symmetrical relation with itself:

```
<attribute name="synonym" valuetype="ReciprocalLink(synonym)" />
```

Apart from these in-built value types, there is the option of defining a set of possible value tokens, called a `ValueSet`. You specify these values with the `value-el` tags (you will see an alternative way right below):

```
<attribute name="emphasis">
  <value-el>strong</value-el>
  <value-el>moderate</value-el>
  <value-el>reduced</value-el>
</attribute>
```

Defining your own ValueSet type Imagine you want to re-use a `valueset` like the one above for the attribute "emphasis" because you have a second track containing the same attribute. Then you can define your own value type in the head section of the specification file. Put all your definitions within the `valuetype-def` tags like this:

```
<head>
  <valuetype-def>
    <valueset name="emphasisType">
      <value-el>strong</value-el>
      <value-el>moderate</value-el>
      <value-el>reduced</value-el>
    </valueset>

    ...

    <valueset name="...">
      ...
    </valueset>
  </valuetype-def>
</head>
```

Now the definition of the emphasis attribute becomes much more compact:

```
<attribute name="emphasis" valuetype="emphasisType" />
```

Setting default values When adding a new track element using Anvil's graphical user interface (Section 4), the following default values will automatically appear in the attributes:

value type	*default*
String	empty
Boolean	"false"
MultiLink	empty
ReciprocalLink	empty
ValueSet	"none"

If you want another default value for your attribute, specify this in the `attribute` tag like in the following examples:

```
<attribute name="foo" valuetype="String" defaultvalue="hey joe!" />
<attribute name="weather" defaultvalue="sunny">
  <value-el>rainy</value-el>
  <value-el>sunny</value-el>
</attribute>
<attribute name="repetition" valuetype="Boolean" defaultvalue="true" />
```

Suppressing "none" (empty value) In a ValueSet attribute the coder usually has the option to select "none" instead of one of the user-defined tokens. This "none" is the so-called empty value. You can tell Anvil not to accept empty values by setting `emptyvalue` to "false". This makes most sense in conjunction with a default value:

```
<attribute name="weather" emptyvalue="false" defaultvalue="sunny">
  <value-el>rainy</value-el>
  <value-el>sunny</value-el>
</attribute>
```

Using graphical symbols An attribute ValueSet can also contain graphical symbols (icons) that you must provide as graphics files (e.g. in JPG or GIF format). To use this feature, first create a subdirectory in the Anvil directory and call it "usericons". Copy all the graphics files you want to use there. Then, define for each value which graphics you want to attach to it. **Note that you also must provide a normal value name!** Using only graphics is not possible at the moment. See the following example as a guideline:

```
<attribute name="weather">
  <value-el image="rainpic.jpg">rainy</value-el>
  <value-el image="smiley.gif">sunny</value-el>
  <value-el>cloudy</value-el>
</attribute>
```

As you can see above, you do not have to attach an image to every value.

Attribute inheritance As mentioned before, the attribute definitions belong to a `track-spec` block. They can also be used in a `group` block. Although a group itself does not contain elements, the specified attributes are *inherited* by all tracks subordinated to the group:

```
<group name="gesture">
  <attribute name="phase">
    <value-el>preparation</value-el>
    <value-el>stroke</value-el>
    <value-el>retraction</value-el>
  </attribute>
  <track-spec name="deictic" type="primary">
     ...
  </track-spec>
  <track-spec name="beat" type="primary">
```

```
    ...
  </track-spec>
</group>
```

In the example we defined two tracks in the "gesture" group. Remember that the tracks' names will be "gesture.deictic" and "gesture.beat". Because of attribute inheritance both tracks will have an attribute "phase" with the specified ValueSet.

8.5 Display Options

Display attributes On the annotation board, elements are displayed as boxes. The contents of the elements consists of attributes and values, which ideally you would like to see all in this box. However, due to the limited amount of space in such a box, you can select which (most important) attributes you want to be displayed directly on the board. We call these attributes *display attributes*. The current values of all display attributes will show up in the box, separated by commas. Empty attributes will not be displayed at all. To make an attribute a display attribute just add `display="true"` to the `attribute` tag:

```
<track-spec name="phrase" type="span" ref="gesture.phase">
  <attribute name="category" valuetype="gestureType" display="true"/>
  <attribute name="emblem type" valuetype="emblemType" />
   ...
</track-spec>
```

Link Color-coding When using links (value types "MultiLink" or "ReciprocalLink") you can let Anvil display the linked-up elements in a specific color (for a list of colors look at Appendix B). You specify the color like this:

```
  <attribute name="my link" valuetype="MultiLink" link-color="orange" />
```

Element Color-coding Color-coding can be used to display the value of a single attribute in terms of colors. For this, you have to two things: First, specify the attribute you want to color-code using the `color-attr` feature in the `track-spec` tag:

```
<track-spec name="trl" type="primary" color-attr="emphasis">
  ...
</track-spec>
```

Second, specify the colors for the different values. Anvil offers a number of standard colors like "green", "yellow", "gray" etc. You can find a complete list of colors in Appendix B:

```
<track-spec name="trl" type="primary" color-attr="emphasis">
  ...
  <attribute name="emphasis" valuetype="emphasisType">
```

```
      <value-el color="red">strong</value-el>
      <value-el color="light red">moderate</value-el>
      <value-el color="orange">reduced</value-el>
    </attribute>
  </track-spec>
```

Arbitrary colors can be defined with an RGB value in hex format:

```
      <value-el color="#eeff00">strong</value-el>
      <value-el color="#ffaa00">moderate</value-el>
      <value-el color="#aaff00">reduced</value-el>
```

Font and font color selection You can change the fonts (including color) used on the annotation board. By inserting a font specification within the confines of the body tag you define a font for the whole annotation board (all tracks).

```
<body>
  <font face="SansSerif" style="bold" color="red" size="9" />
  ...
</body>
```

By inserting a font specification within a track-spec tag you define a font for only the respective track, overriding other font specifications.

```
<body>
 ...
 <track-spec>
  <font face="SansSerif" style="italic" color="blue" size="10" />
  ...
 </track-spec>
</body>
```

Note that you need not specify all features (e.g. you can omit color or size – Anvil will fall back on default values for such cases). For possible color values look at Appendix B. For "style" you can specify: plain, italic, bold or bolditalic. For "face" you can try:

- Dialog
- DialogInput
- Monospaced
- Serif
- SansSerif

8.6 GUI Customization

Element Window The temporal borders of track elements are showed in the top section of the Element Window. You may need a more precise display than given in the standard case (i.e. hh:mm:ss). Anvil can be made to display seconds (up to four decimals) by putting the following into the `head` section of your specification:

```
<gui-settings>
  <track-window precise-time="true" />
</gui-settings>
```

8.7 Documentation

If you add documentation to your specifications, this information can be accessed in two ways. First, while coding you can always look into your definitions by clicking “info” buttons (see Figure 9, page 17). Second, you can let Anvil generate a complete coding book in HTML that can be browsed with an Internet brower like Netscape or Internet Explorer (Section 5.11).

Documentation can be added for groups, tracks, sets, attributes and even attribute values. These explanations must be bracketed in `doc` tags and can contain HTML formatting tags:

```
<track-spec name="posture shift" type="primary">
  <doc>
    This track registers any movement in (sitting) posture like
    shifting on the chair, crossing/uncrossing legs etc.
  </doc>
  <attribute name="movement" >
    <doc>
      Code the <b>type</b> of movement here. Start at the first frame where
      movement can be seen. End where all body parts are in rest
      position.
    </doc>
    <value-el>
      cross-legs
      <doc>
        Subject crosses legs.
      </doc>
    </value-el>
    <value-el>
      ...
    </value-el>
    ...
  </attribute>

  <attribute name="certainty" valuetype="Number(0,4)">
    <doc>
      How certain are you of your coding?
    </doc>
  </attribute>
  ...
```

9 Other Annotation Projects

There has been a lot of research done in linguistic and, more recently, in multi-modal data annotation. In this section I will briefly review some projects that are related to Anvil. One has to distinguish between annotation *schemes*, annotation *frameworks*[12] and *tools*. Tools can be further differentiated by the tasks they can handle: annotation/coding, data retrieval, analysis etc. Not all of the reviewed projects cover all aspects.

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 that supports multiple tracks. There is one reference track which all other tracks refer to, the so-called canonical track which contains words in a phonological transcription. The elements of other tracks (lexical entries, dialogue acts, part-of-speech etc.) can point to an arbitrary collection of elements of the canonical track. The description of an element is a simple string. Partitur is actually just a *file format*, meant for centralized data collection/distribution. There are no generic tools for coding.

The completed European project **MATE**[13] (Multilevel Annotation Tools Engineering, 1998–2000) aimed at producing a generic workbench for linguistic coding and data retrieval. Annotation schemes from different fields and projects were taken as models, different file formats (e.g. BAS Partitur) are supported in an XML-based approach. Although powerful retrieval concepts were developed, the tool as a whole is not available in a stable version where coding can be practically pursued.

The ongoing US project **ATLAS** (Architecture and Tools for Linguistic Analysis Systems, cf. Bird et al., 2000b) introduced the concept of *annotation graphs*. A large number of schemes are compatible with annotation graphs. Anvil's object model bears some similarity to annotation graphs. In ATLAS, much work has been done by Bird et al. (2000a) for finding retrieval mechanisms which do not exist yet in Anvil. The data exchange format is also kept in XML.

CAVA[14] (Computer Assisted Video Analysis) is a system of tools, developed at the MPI for Psycholinguistic at Nijmegen. It offers tools for annotation, retrieval and analysis. CAVA's components require different platforms (Mac, PC) and use special hardware which makes it difficult to install. A component comparable to Anvil is the Mac-based *MediaTagger* used for annotation of Quicktime files. It is currently being replicated in an extended,

[12] I take an annotation framework to be a representation mechanism for the entities of an annotation scheme (cf. Bird and Liberman, 2001). 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.

[13] `http://mate.nis.sdu.dk`

[14] `http://www.mpi.nl/world/tg/CAVA/CAVA.html`

Java-based version called **EUDICO** (European Distributed Corpora), a European project (Brugman et al., 2000) still in progress. EUDICO's coding tool ELAN is already available.

The **TASX**[15] (Time Aligned Signal data eXchange) project is an annotation project at the University of Bielefeld, Germany. Its goal is to create an environment for the storage, exchange and annotation of linguistic data. A Java-based coding tool, the TASX-Annotator, is already available. It offers a number of different views on the data: the time-aligned view (similar to Anvil's annotation board), the score view (HTML, non-editable) and the text view (one element per line). All data is stored in XML. XSL is used for data conversion. Conversion tools already exist, also for Anvil data.

The **MMAX**[16] tool was developed at the European Media Lab (EML), Heidelberg, for the annotation of multi-modal corpora (Müller and Strube, 2001). It offers an annotation framework that was inspired by MATE: elements of different types can contain attributes and point to each other. Types are comparable to Anvil's tracks, though the type concept allows overlap and nesting within one track. Elements are called *markables*. The tool is written in Java with JMF, is XML-based and meant to be scheme-independent. The graphical user interface only supports the mark-up of *texts*, there is no video component integrated.

Akira is a video analysis tool developed at the University of Mannheim for film studies and allows multi-track annotation with an intuitive user interface. 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.

MacSHAPA[17] is a Macintosh software for "observational data analysis" developed by Penelope Sanderson at the University of Illinois (Sanderson et al., 1994) that allows transcription with your own annotation scheme. To my knowledge, it is not developed or maintained any more but one can still get a software copy and the 800-page manual that comes with it. MacSHAPA offers the following statistical analyses: content analysis, duration analysis, transition analysis, lag sequential analysis, reliability (kappa) and cycle reports.

The Observer (version 4.0) by the Noldus company[18] is a *commercial* system. It allows multi-layered annotation by the concept of classes (which losely correspond to Anvil's tracks) and behavioral elements (Anvil's track elements). Annotation is done on a *checksheet* where coded elements appear in temporal order like in a list. It does *not* offer a time-aligned view while working on the annotation which is probably its greatest drawback. On the

[15]`http://coli.lili.uni-bielefeld.de/~milde/tasx/`
[16]`http://www.eml.org/english/research/NLP`
[17]`http://www.it.swin.edu.au/projects/macshapa`
[18]`http://www.noldus.com`

analysis side, The Observer offers simple descriptive stats (frequencies), reliability studies and lag sequential analysis (transitions and transition times). Only here can the time-aligned view be computed. The Observer is a robust, professional tool but has some restrictions in the definition of behavioral elements (only two attributes per element), it is not platform-independent (MS Windows only) and not XML-based.

10 Acknowledgements

I would like to thank a number of people for precious input on bugs, missing functionality and technical solutions (listed in alphabetical order, hope I did not forget anyone): Stephan Baldes (DFKI), Piero Cosi (CNR), Ralf Engel (DFKI), Jennifer Cornish (U Buffalo), Elena Frigerio (U Milan), Patrick Gebhard (DFKI), Emanuela Grandinetti (U Milan), Lisa Harper (MITRE), Jeffery Higginbotham (U Buffalo), Martin Klesen (DFKI), Christoph Lauer (DFKI), Dan Loehr (MITRE), Jean-Claude Martin (LIMSI), Brian MacWhinney (CMU), Jan-Torsten Milde (U Bielefeld), Kari Anne Myers (LDC), Chris Osborn (LDC/UPENN), Norbert Reithinger (DFKI), Graziano Tisato (ISTC), Hannes Vilhjalmsson (MIT), various other people from the LDC at UPENN, and all partners of the European NITE project.

References

Alexandersson, J., Engel, R., Kipp, M., Koch, S., Küssner, U., Reithinger, N., and Stede, M. (2000). Modeling Negotiation Dialogs. In Wahlster, W., ed., *Verbmobil: Foundations of Speech-to-Speech Translation.* Berlin: Springer. 441–451.

Bird, S., and Liberman, M. (2001). A Formal Framework for Linguistic Annotation. *Speech Communication* 33(1–2):23–60.

Bird, S., Buneman, P., and Tan, W.-C. (2000a). Towards a query language for annotation graphs. In *Proceedings of the Second International Conference on Language Resources and Evaluation*, 807–814.

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

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

Kipp, M. (2001a). Analyzing Individual Nonverbal Behavior for Synthetic Character Animation. In Cavé, C., Guaïtella, I., and Santi, S., eds., *Oralité et Gestualité. Actes du colloque ORAGE 2001*, 240–244. Paris: L'Harmattan.

Kipp, M. (2001b). Anvil – a Generic Annotation Tool for Multimodal Dialogue. In *Proceedings of Eurospeech 2001*, 1367–1370.

Kipp, M. (2001c). From Human Gesture to Synthetic Action. In *Proceedings of the Workshop on "Multimodal Communication and Context in Embodied Agents" (Agents-2001)*, 9–14.

Mann, W. C., and Thompson, S. A. (1988). Rhetorical Structure Theory: Toward a functional theory of text organization. *Text* 8(3):243–281.

Martin, J.-C., and Kipp, M. (2002). Annotatng and Measuring Multimodal Behaviour - Tycoon Metrics in the Anvil Tool. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC).*

Müller, C., and Strube, M. (2001). MMAX: A tool for the annotation of multimodal corpora. In *Proceedings of the 2nd IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, 45–50.

Sanderson, P. M., Scott, J. J. P., Johnston, T., Mainzer, J., Watanabe, L. M., and James, J. M. (1994). MacSHAPA and the enterprise of Exploratory Sequential Data Analysis (ESDA). *International Journal of Human-Computer Studies* 41:633–668.

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*.

van Lieshout, P. (2000). Praat workshop. a basic introduction. Technical report, University of Toronto.

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

A Features of Anvil 4.0

New features with regard to the former version are printed boldface.

- General
  - plug-in facilities
  - platform-independence (only Java 2 and JMF required)
  - frame-accurate video control (and slow motion)
  - display of waveform
  - multi-layered annotation in hierarchical tracks
  - track elements are objects with attribute-value pairs
  - attributes can hold arbitrary links to other elements
  - links are color-coded
  - links can be reciprocal
  - all tracks/attributes are completely user-defined
  - special container type "set" for non-temporal objects
  - hotlist (direct access to most recently opened files)
  - customizable window layout
  - auto-layout
- Viewing and Editing
  - playback of marked segment
  - elements can be cut and extended after insertion
  - free-form comment in elements (marked with flag)
  - bookmarks (like in Internet browsers)
  - automatic generation of coding manual HTML pages
  - collapsible group nodes (view)
  - scalable track height
  - color-coding of track elements
  - context popup menus
  - element forward/backward hopping
  - jumping to time/frame
  - online attribute documentation in edit window
  - group collapsing/expanding on double click
  - mouse wheel for scrolling (due to Java 1.4)
  - comment display can be switched off (view option)

  - **configurable fonts for the annotation board**
  - **graphical symbols for annotation**

- File Exchange
  - XML format for all file exchange
  - import interface for RSTtool
  - import interfaces for PRAAT and XWaves
  - import of pitch data from PRAAT (linked up as file)
  - import of intensity data from PRAAT
  - export to text files
  - export to text tables (for stats packages like SPSS or Statistica)

- Search and Find
  - global search across annotations (Project Tool)
  - click and jump connection from hitlist to Anvil

- Analysis and Presentation
  - print-out of annotations (not reliable!)

B Anvil Color Tables

Elements can be color-coded according to their values in a ValueSet as outlined in Section 8.5 and shown in a small example:

```
<track-spec name="trl" type="primary" color-attr="emphasis">
  ...
  <attribute name="emphasis" valuetype="emphasisType">
    <value-el color="red">strong</value-el>
    <value-el color="light red">moderate</value-el>
    <value-el color="orange">reduced</value-el>
  </attribute>
</track-spec>
```

Possible value colors are:

	black	
light blue	blue	dark blue
light cyan	cyan	dark cyan
light gray	gray	dark gray
light green	green	dark green
light magenta	magenta	dark magenta
light orange	orange	dark orange
light pink	pink	dark pink
light red	red	dark red
	white	dark white
light yellow	yellow	dark yellow

Links can be color-coded as outlined in Section 8.5 and shown here:

```
<attribute name="my link" valuetype="MultiLink" link-color="orange" />
```

Possible link colors are:

green
red
magenta
orange
silver
blue
yellow

C Specification File Sample

This is an excerpt of a specification file (extension “xml”), containing the specification of the “trl” (transliteration) track and the “deictic” track. Note that attribute value sets can be defined in the head as so-called ValueSets and assigned a name (see “phaseType” in the excerpt). Thus, the same set of values can be reused in different tracks.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<annotation-spec>
 <head>
  <valuetype-def>
   <valueset name="phaseType">
    <value-el color="#eeee00">
      prep
    </value-el>
    <value-el color="#dd0000">
      stroke
    </value-el>
    ...
   </valueset>
  </valuetype-def>
 </head>

 <body>
  <track-spec name="trl" type="primary" color-attr="emphasis">
    <doc>
      This track contains the transliteration of the spoken
      discourse. The <b>token</b> attribute contains the word or sound
      uttered. There are a number of special marks signified by
      squared brackets like [breath] or [aeh] for the mark-up of
      nonverbal sounds. Note that there is <b>no syllable
      information</b>.
    </doc>
    <attribute name="token" display="true">
      <doc>
        Words and nonverbal sounds.
      </doc>
    </attribute>
    <attribute name="emphasis" valuetype="emphasisType">
      <doc>
        Emphasis of this word. Possible values are the same as in
        SABLE.
      </doc>
    </attribute>
  </track-spec>

  ...

  <track-spec name="deictic" type="primary" color-attr="phase">
   <doc>
     The good, old <b>pointing gesture</b>. A pointing gesture can
     be directed at a concrete entity (person, place, object) or at
     something abstract (pointing into gesture space to refer to
     something said before or to refer to a distant place,
    e.g. another country/university). Most interesting attribute
    is <b>where</b>.
  </doc>
  <attribute name="phase" valuetype="phaseType">
    <doc>
       My phase description is based on the phases postulated by
       Kendon and McNeill (1992), later on extended by Kita et
       al. (1999).
    </doc>
  </attribute>
  <attribute name="where" display="true">
    <doc>
       The object/aim of the pointing gesture.
    </doc>
    <value-el>
       addressee
       <doc>
         Subject points to addressed person/object.
       </doc>
    </value-el>
    <value-el>
       space
       <doc>
         Subject points somewhere in gesture space, points to
         nothing concrete that is present in the room.
       </doc>
    </value-el>
    <value-el>
       self
       <doc>
         Subject points to him/herself usually by putting the hand
         on his/her chest.
       </doc>
    </value-el>
    <value-el>
       2directions
       <doc>
         Points to two different directions with his/her two hands,
         e.g. to himself and to addressee.
       </doc>
    </value-el>
  </attribute>
 </track-spec>

 ...

</body>
```

D Annotation File Sample

This is an short excerpt of a sample annotation to show the syntax of an anvil annotation file (extension “anvil”).

Note that the information of `type` and `ref` in the `track` tag are obsolete because these are specified in the associated specification file (these features are a relic from former times when there was no specification file — left them because they are possibly useful for file conversion tools). Anvil will not complain if this information is falsely specified in this file but will write the correct data back when saving the annotation.

```
<?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>
```