# AVATecH Recognizer Interface Specification Manual

Version 2.1, 6 March 2014

E. Auer, H. Sloetjes, P. Wittenburg, D. Sørensen

## Contents

**Tables of figures**

# Changes in the AVATecH Recognizer Interface Specification manual since version 1.0

Updates generally follow the rule that old software stays compatible: Additions can be absent in or ignored by old software. All sorts of new formats are additions to the old formats (or protocol keywords or other details) so when **a user** use**s** old recognizers in new ELAN or vice versa, newer features **are not in the way** but mostly just ignored.
For example when you think about progress information, parameters to store invocation context, attributes to mark parameters as advanced and numbers as integers - they all add functionality**,** but old software will not be confused when encountering them. It will just not make full use of the possibilities**.**
A good example is the level=”basic” or level=”advanced**”**. Each parameter item in the CMDIs (of any type, including input and output files) can be marked as level="basic" or level="advanced". If a CMDI does not specify whether a recognizer option is for advanced users, it is assumed to be for basic.

It is now possible to have multiple cmdi files in the same folder in the extensions folder, the name recognizer.cmdi is no longer mandatory.

# Introduction to AVATecH

AVATecH is meant to deliver sound and video pattern detection recognizers that are producing annotations which then can be used during the analysis process. It must be possible for users to invoke these recognizers interactively from a caller software which can be:

- ELAN for the selected file or
- ABAX batch program that allows to execute the operation on a whole set of files.

These recognizers can appear in different forms dependent on the complexity of the execution framework that they require:

- There are "simple" recognizers that operate as plug-ins and
- There are "complex" processing frameworks that operate in their own realm; some of these processing frameworks even require certain operating systems and it would be far too costly to modify the code to run on all platforms.

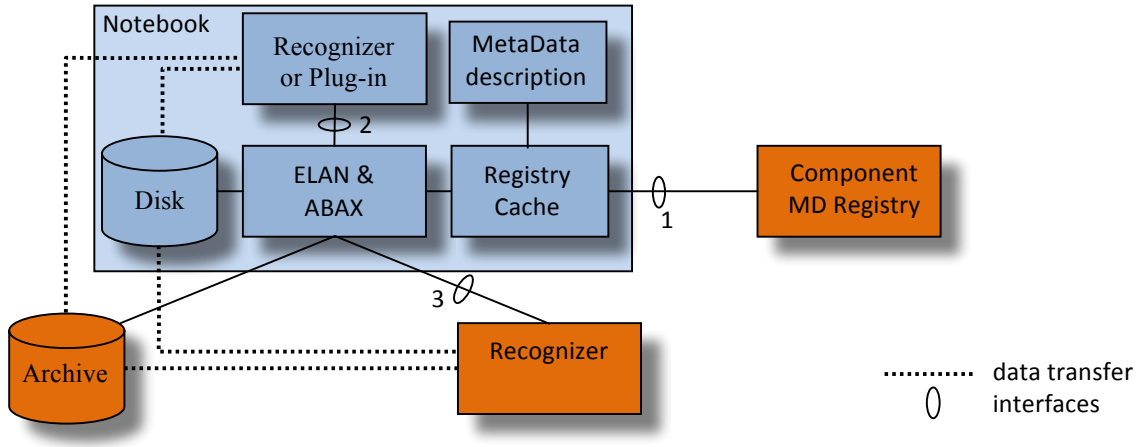



*Figure 1: Overview of AVATecH*

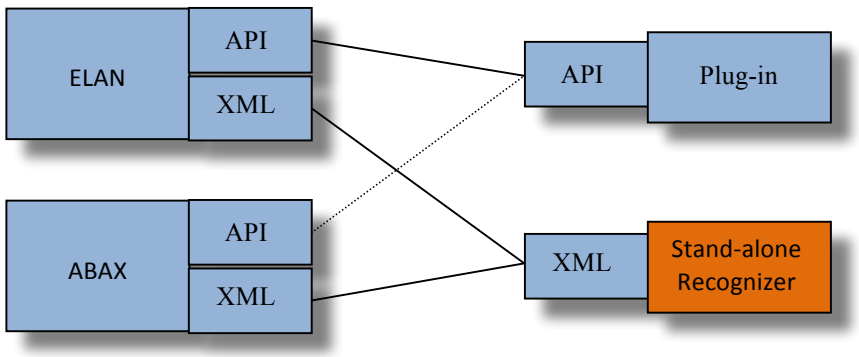
AVATecH must cater for all situations that are indicated. It is important that the user will be confronted with similar user interfaces to not have to learn too many different things, i.e. ABAX and ELAN need to behave similarly. It should be mentioned here that the caller functionality will be included in ELAN and the ABAX stand-alone batch tool that can be executed on a notebook or a server. The complexity of complex operations should be hidden behind a simple and known user interface and where possible ELAN should be the entry point in general. Since ELAN is running on a notebook (MAC, Linux or Windows environment) we also need to consider that some recognizer programs only work on dedicated servers, i.e. we need to be aware of the fact that some commands need to be executed in a local area network. But we assume that the components have access to one shared file system. Yet we will not anticipate the Internet as an execution environment. Recognizers can be installed on notebooks or on servers dependent on their characteristics.

# Interfaces

1. The caller, be it ELAN or ABAX, will interact with a recognizer metadata registry (collection of recognizer metadata sets). Metadata is available on a server at MPI and locally to the caller. All metadata descriptions are made according to a CMDI component specification. For being able to work locally a cache is maintained on the notebook which is synchronized while being connected to the LAN.

2. An API is specified that exactly describes how a plug-in will need to be programmed and how it will be invoked. Data transfer will in general be done from a shared network drive.

3. Since we assume that for self-standing recognizers the processing time in general will be high we can ignore the efficiency of the interfacing. Therefore it is preferred to exchange XML streams which a) send all information that a recognizer requires and b) receive progress and state reports. In both cases data files (input data, output data) are specified by URLs or paths so that the recognizer can directly read/write from/to files. The XML syntax of both files and the execution and control options has been specified so that recognizer developers know exactly how to offer them.




Both callers will finally realize the two interface types (direct API and XML based exchange). Ways to reuse code will be studied, but this is too early. With respect to the priorities we can say that ELAN will now focus on the implementation of the API interfacing, while ABAX will focus on the XML interfacing.

*Figure 2: The interaction possibilities intended*

# User Interfaces

When working on a single file and testing the options of recognizers in a kind of interactive mood we assume that users will first start ELAN and look at the A/V streams and the annotations. We assume that they then will start a recognition process be it implemented as plug-in or as self-standing recognizer which will lead to new annotations and time series which ELAN should be able to present in a suitable way. Thus the interaction occurs before and after the execution of a recognizer. While a plug-in allows more direct access to ELAN features, stand-alone recognizers can run more easily on remote servers or be written in other programming languages.

However, we foresee situations where recognizers may need an interaction while processing. Yet we cannot describe a useful scenario. In this case it was agreed that the recognizer would need to use its own graphic interaction methods, since only the recognizer will be able to generate the information streams and will be able to map the user responses. This may restrict the possible configurations, since it may be required that the recognizer is able perform its interaction by using a specific display.

# 1. Recognizer Selection

The user will in general start working from ELAN as the initial software framework. The user must then be able:

- to first select one or more files to work on and
- to second select a processing recognizer which is either a plug-in or a recognizer that is being invoked via an API

A recognizer is described by proper metadata independent of its type, thus they share the same description elements. However one element needs to tell ELAN whether the recognizer is a plug-in or self-contained recognizer. ELAN then knows what kind of invoking option to use and what kind of information to be requested.

# 2. Recognizer Execution

## 2.1 Case A - Interactive Use

1. The user will start ELAN and open a certain file (file bundle).
2. Then the "Recognizer" option in ELAN will be selected using the selected file.
3. The user then is looking for recognizer that can work on the currently open files.
4. A recognizer will be selected and the appropriate interactions will be started to specify parameters and additional resources.
5. All information will be packaged end sent to the recognizer. Here a difference will be made between a plug-in and a self-standing recognizer perhaps even running on another server in the LAN.
6. The recognizer will be started and will provide some results. Again a difference will be made between a plug-in and a recognizer. In the second case a file will be used to exchange the results without including the data streams. This file will include pointers to resulting annotation or time series files.
7. ELAN will be able to visualize the newly created annotations and time series.

## 2.2 CASE B - Batch Use

1. The user will start the stand-alone ABAX batch tool.
2. Then the files to be processed will be selected.
3. A script file selects recognizers that can work on the set of files.
4. The appropriate parameters and additional resources are specified in the script file or a preferences file.
5. A loop option will be started to process one file after the other.
    a. All information will be packaged end sent to the recognizer. Here a difference will be made between a plug-in and a self-standing recognizer which could even be running on another server in the LAN.
    b. The recognizer will be started and will provide some results. Again a difference will be made between a plug-in and a recognizer. In the second case files will be used to receive the results, such as annotation or time series files.
6. The batch tool will collect the result files.

To support the user, a wizard tool can help with the creation of script and preference files. This tool gathers information about which recognizers will use which parameters and which recognizers will be called in which order. It can also help defining chains, where output of one recognizer is used as input for the next recognizer.

# 3. Interface Specifications

It is agreed that any interaction which is not based on direct program-to-program communication but requires an intermediate representation needs to be in XML based on an agreed schema.

## 3.1 Recognizer Interface Specifications

### 3.1.1 Metadata Description

Both resources and recognizers (whether implemented as plug-ins or as self-standing components) are described by metadata descriptions. Amongst others it specifies what kind of task the recognizer can carry out and what types of files it can read and write. It also needs to have information how a recognizer can be started. The format of this file belongs to the MPI CMDI component metadata family of XML file formats and is described in detail in the next sections.

### 3.1.2 Overview of Interaction

The steps involved in the interaction between the user, ELAN/ABAX, metadata registry, recognizer components and hard disk are outlined in the diagram below. If the caller is a batch tool, user interaction should be replaced by looking up values from the batch file or (optionally) an INI file which stores user specific preferences. As indicated in the above session there are differences with respect to the way for example how the interaction is implemented for plug-ins and self-standing components. Also the disk can reside at different places.

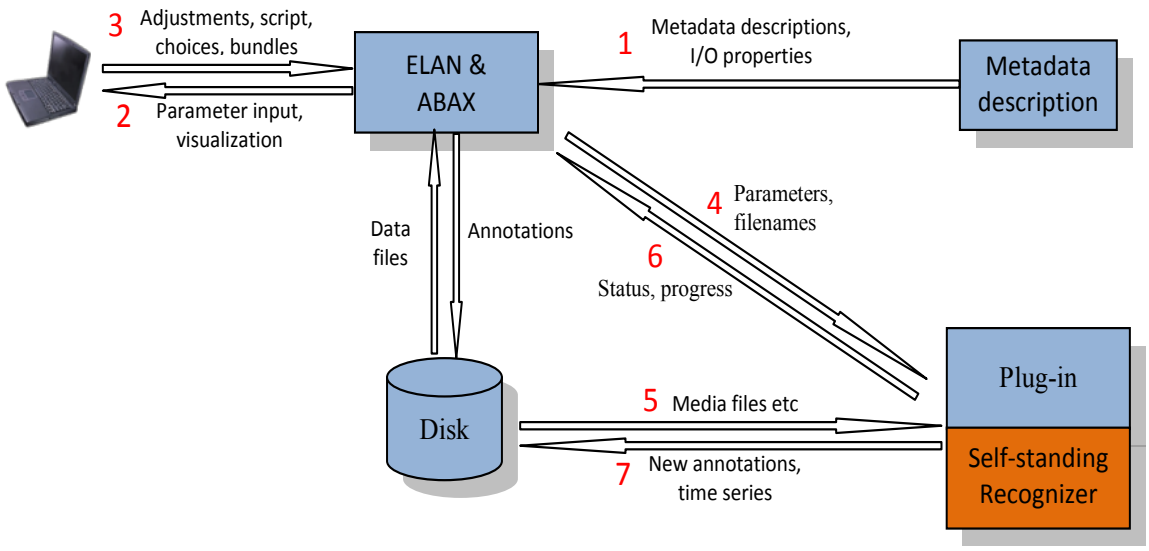



*Figure 3: Overview of interaction (red numbers indicate sequence of steps)*

### 3.1.3 Detection *[Step 1]*

As indicated, it could happen that the user first selects a resource bundle and then decides to do some operations on it. This would change the steps slightly. We will now discuss **the alternative path** where the user first selects a recognizer to be executed.

In the metadata registry which is a local cache on the machine ELAN or ABAX are operating on, descriptions of all recognizers (plug-in or self-standing component) can be found. The interaction module will read all *recognizer.cmdi* files from that registry. This metadata gives information about name, description, binding information, runtime parameters, data input and data output properties of the recognizer. An interactive software such as ELAN will typically collect name and description of all accessible recognizers and show them in a menu for selection, while ABAX, a batch scripting tool, will use a script file to find all information which needs to be specified. Since both frameworks will be used as entry points for the user, both could include the same interaction module.

The metadata can specify different ways of calling the recognizer depending on the operating system of the caller. It is possible that a recognizer does not support the system the user is working on. This will not be a problem for "shared" recognizers invoked via the LAN.

For a more detailed documentation, there is a HTML file linked from the CMDI file. This HTML ships with the recognizer and gives extra explanations for using the recognizer and information about licenses, authors and contacting them, version numbers etc. The link can be either a relative link to a file in the same directory tree as the CMDI file or (not recommended) a direct http link to a recognizer homepage elsewhere. The link URL is stored as the value of the 'documentation' element of the CMDI. In an integrated environment like ELAN, the link could be opened in a simple embedded HTML viewer, with a possibility for the user to open the link in a browser. The documentation page should avoid too complex HTML, to make embedded viewing easier. In particular old CMDI files are likely to have no documentation link. ELAN could disable the corresponding menu item in that case, or show a message and the raw contents of the CMDI file instead to let the user investigate the problem.

Recognizers have to specify a **recognizerType** in their metadata, which can be either *direct* (using direct Java invocation, as described in the appendix), *local* (using local files and XML for parameters and feedback), *shared* or *remote*. In the latter two cases, the recognizer itself runs on another computer. However, in the shared case, recognizer and caller share one file system, for example in a local intranet.

Our example *recognizer.cmdi* file describes a recognizer running on the local Linux computer of the caller, with the name "TagVowels", which for example in Linux can be invoked by running the file "PitchPraat.sh". The directory is relative to *cmdi*, in this case PitchPraat.sh and *recognizer.cmdi* are in the same directory. The description is "Tag vowels (volume peaks of voiced spans)".

**Local** recognizers have to specify local commands to run them, per operating system. The recognizers communicate via XML streams and via files, as described later.

**Direct (Java)** recognizers use the same **runLinux**, **runMac** and **runWin** attributes to specify the fully qualified name (also known as the binary name) of the class implementing the Recognizer interface. For this option see 3.2 Plug-In Interface Specifications.

**Shared** filesystem recognizers have to specify URLs which can be contacted for sending and receiving the parameters and feedback XML streams. Which URL is used depends on the caller operating system, to be more exact on the file name syntax used for the parameters. If the URL uses **telnet** protocol, parameters and feedback are sent and received as raw XML data via TCP/IP. If the URL specifies **http** as protocol, parameters must be sent as REST GET or POST form data via HTTP and the XML feedback must be sent as HTTP content.

**Remote** filesystem recognizers could be a future extension and are not supported yet. The caller will have to specify the input file locations by URLs as seen from the recognizer server. Output file locations could be URLs inside a web page hosted by the recognizer server, or specify some sort of upload to the caller computer. Maybe HTTP multipart forms could be used. Fully REST based invocation of recognizers already is possible with the existing protocol with help of a small client proxy: In the sample implementation, the client proxy simulates a local recognizer, but uploads files to a temporary workspace of a CLAM (http://ilk.uvt.nl/clam/) wrapped installation of the actual recognizer on a remote server, via REST. It then sends parameters, receives logs, downloads result files and closes the workspace, all invisible to the user who can treat the proxy as if it were the actual recognizer.

### 3.1.4 Fetching run-time settings *[Step 2]* / *[Step 3]*

In the next step, the metadata is used to determine which information is needed to run the recognizer. This information is then either fetched from a configuration file (batch case) or requested interactively from the user (interactive case).

In the batch case, the caller batch tool could for example process an INI file listing the values for each of the parameters (by parameter name) of each of the invoked recognizers (corresponding to sections in the INI file).

In the interactive case, the caller will parse the metadata and present an input form (e.g. via a GUI) with suitable input fields to the user. Callers can implement custom forms for some

frequently used recognizers for added comfort, as long as collected values are within the specs defined by the metadata.

### 3.1.5 Parameters

Recognizers can have any number of textual or numerical runtime parameters. Each parameter has a name, a description and a default value. Each parameter item in the CMDIs (of any type, including even input and output files) can be marked as level="basic" or level="advanced". If a CMDI does not specify whether a certain option is for advanced users, it is assumed to be for basic users. Software which helps the user to run recognizers can have an option to hide all advanced settings of recognizers from the user interface, possibly with a default of not showing advanced settings until the user enables showing them.

**Numerical parameters** in addition always have a minimum and maximum allowed value. Recognizers can decide to use only the integer part of a parameter, but the caller is free to send floating point values for all numerical parameters, in non-scientific notation (no exponent).

The CMDI 'numparam' element can use a 'type' attribute which can be either 'int' or 'float' to specify the syntax preferred for a parameter. Sending non-integer values for integer parameters is not an error, but should be avoided. The recognizer can either round or truncate the value to an integer. Sending integers for float parameters is always allowed, recognizers have to be able to parse either format.

**Textual parameters** have an optional "controlled vocabulary". If a textual parameter does not have a controlled vocabulary, any line of text can be used as value. Recognizers may ignore leading or trailing whitespace in textual parameters.

If a controlled vocabulary is given and non-empty, any one of the (space separated) words in this vocabulary can be sent as parameter value; this can be used for choices and booleans.

The example *recognizer.cmdi* file describes a text choice parameter called "loglevel" which can have either of the three values "normal", "verbose" or "debug", default being "normal". It also defines a numerical parameter called "max_f0" which must be between 150 and 1500 (inclusive) and has a default of 550. The example *recognizer.cmdi* file also defines parameters called "epsilon", "vol_threshold" and "voicedness_threshold".

### 3.1.6 Input data and file formats

A recognizer can have any number of input files taken from the following five categories: tier, multitier, timeseries, audio, video and auxiliary. Many recognizers will take only one input file and this input file will often be audio or video. Tier and timeseries are also available as CSV items instead of the default XML, while the more complex multitier format is not.

Recognizers can use any combination of input files and formats. For example video recognizers can use audio for extra knowledge about what they "see" and if a good uncompressed WAV audio is available for a bad mpeg video, why not use it? Also, recognizers can be used nicely to make tiers (or other data) from other tiers, for example they

can merge or modify tiers according to some algorithm. Even some recognizer with only buttons and no files as input could be useful, e.g. to make random annotations or if there is some external data source outside the annotation environment, such as a live measurement..
All input files will be sent to the recognizer like a "parameter" with the file name as the value. The syntax of file names will depend on the used operating system. A recognizer might also allow processing of remote files given by URL, but for large audio or video files this cannot be recommended and callers do not normally have to provide user interfaces to pass URL values.

Input (and output) files can have their supported mimetypes specified: For example audio input can be specified further in the metadata file to be allowed to be WAV or MPEG4 audio files, by setting the **mimetypes** attribute to **audio/x-wav audio/mp4** (space separated). XML encoded tiers or timeseries will be **text/xml** and CSV will be **text/csv**. For generic audio and video the mimetypes will be audio/* and video/*. The caller can use this information as hint in selecting input files of the right type and to know which output file formats to expect. Recognizer and caller still have to process all file headers (for codec and format choice etc) so the mimetypes lists in the metadata are only support data.

**In CSV formats,** time, start time and end time columns in tier and timeseries files can have arbitrary names. Those columns must be first (leftmost) but their names are not relevant.The times are in units of seconds, in 42.075 style syntax. No hh:mm:ss or exponent syntax allowed.

To cancel a recognizer run, abort LOCAL recognizer process (Ctrl-C, SIGINT, Process destroy() etc), disconnect from SHARED recognizer or use stop() method of DIRECT recognizer.

**Timeseries** must be encoded in XML as shown in the example below. Non-ASCII characters must be encoded as Unicode UTF-8, < and & must of course be escaped as < and & respectively. There can be one or more values per item, which can be numerical or text. All items (i) must have the same number of values (v). A **columns** header provides titles for the columns, as a space separated string. Numerical columns must have titles starting with the # character. Of course all items (i) must provide values (v) for all columns and the values must be given in the same order as specified in the columns header attribute. Note that any non-numerical string in a numerical field is just treated as **not-a-number**, the exact string does not matter. For human readability, there should be linebreaks at least after the </i> of each item, as shown in the example:

```
<?xml version="1.0" encoding="UTF-8"?>
<TIMESERIES xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="file:avatechtimeseries.xsd"
  columns="#intensity #voicedness">
  <i t="0.012"><v>3.14</v><v>4.13</v></i>
  <i t="0.052"><v>3.15</v><v>n/a</v></i>
</TIMESERIES>
```

**CSV Timeseries** are an alternative encoding of the normal XML timeseries: Again, non-ASCII characters must be encoded as Unicode UTF-8. Values must not contain linebreaks and values must be embedded in double quotes if they contain semicolons. Otherwise, quotes are optional. Quotes themselves can be encoded as double quotes. Columns are separated by semicolons and the first row contains the column titles. Note that the time column also has a

name, which is ignored. Also, note that the numerical #voicedness column contains the value "n/a" in one row to express an undefined numerical value, a not-a-number, as explained already above for XML Timeseries. The data shown above would be encoded as CSV as follows:

```
#timestamp;#intensity;#voicedness
0.012;3.14;4.13
0.052;3.15;n/a
```

The exact format of the values will depend on the needs of the recognizer, although integers and values with decimals after a decimal point will be common. Unparseable values can be used as Not-a-Number to indicate missing data points for numerical values. The items in a timeseries file have to be sorted in ascending time order but do not have to have fixed time steps from item to item.

A **tier** file is similar to a timeseries, but each item describes a timespan, with start time and end time. Item times have to be in ascending order and the timespans must not overlap. They can, however, be back to back. Content can be free text, numerical values, text using a controlled vocabulary or similar. Of course all spans must provide values (v) for all columns and the values must be given in the same order as specified in the columns header attribute. Some recognizers might use only the times of the timespans themselves and not the content, or might for example only treat the length of the content as their input. Here is an example for the XML representation. For human readability, there should be linebreaks at least after the  of each item, as shown in the example:

```
<?xml version="1.0" encoding="UTF-8"?>
<TIER xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="file:avatechtier.xsd"
  columns="gender #estimate">
  <span start="0.012" end="1.052">
    <v>male</v><v>-0.71</v></span>
  <span start="3.177" end="4.144">
    <v>female</v><v>0.82</v></span>
  <span start="4.144" end="5.237">
    <v>unknown</v><v>0.25</v></span>
</TIER>
```

It is recommended that tiers will be used to communicate about timespans of interest to save CPU. The recognizers will take an optional input tier file to specify the time-"regions of interest", i.e. process only the timespans specified in that tier for fast checks. Such selection tiers should be optional, though: If nothing else is specified, the recognizer still processes the whole duration of the input media.

Is should be to emphasized that tiers are not only a way for recognizers to produce output in the form of annotations but also a good way to send data to the recognizers. Thus it is good to make it easy for users to pick existing tiers to send them to recognizers. In some cases, it might also be useful to let the user create temporary tiers to quickly encode data to be sent to the recognizer.

A **multitier** file is similar to a tier file, but it supports multiple TIER elements. Compared to the tier file, multitier files allow to use independent segmentations in different tiers within a

single XML file. In the classic tier format, it is possible to store multiple tiers, but with only one time segmentation which had to be shared by all tiers. Recognizers that wish to communicate more complex sets of annotation tiers can now use one multitier file instead of using several classic tier files.

```
<?xml version="1.0" encoding="UTF-8"?>
<TIERS xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:noNamespaceSchemaLocation="file:avatech-tiers.xsd">
    <TIER columns="voiceGender #spectralFemalenessEstimate">
        <span start="0.012" end="1.052">
            <v>male</v>
            <v>0.71</v>
        </span>
        <span start="4.144" end="5.237">
            <v>unknown</v>
            <v>0.25</v>
        </span>
    </TIER>
    <TIER columns="speechClassification">
        <span start="0.747" end="0.988">
            <v>question</v>
        </span>
        <span start="3.164" end="3.711">
            <v>emphasis</v>
        </span>
        <span start="4.844" end="5.222">
            <v>cough</v>
        </span>
    </TIER>
</TIERS>
```

**CSV Tiers** are encoded similar to CSV timeseries, so the data above would be encoded in a CSV tier as follows. The double quotes are optional here because none of the values does contain a semicolon in this example:

```
"#starttime";"#endtime";"gender";"#estimate"
0.012;1.052;"male";-0.71
3.177;4.144;"female";0.82
4.144;5.237;"unknown";0.25
```

Tiers could be used to mark timespans of interest or to tag timespans for a recognizer which would tag timespans with similar properties with the same tag in the generated output file.

For **audio**, at least uncompressed 16 bit PCM RIFF WAVE (*.wav) files have to be supported, in mono and stereo. Recognizers have some parameter to select a channel of a stereo recording, but the default processing of stereo files will just use both channels and, if necessary, mix them to mono for processing. The mimetypes attribute in the metadata should contain at least the value audio/x-wav which means any variant of RIFF WAVE but the recognizer only has to support the uncompressed variant.

For **video**, at least MPEG-1 and MPEG-2 files with pack header have to be supported (starting with 00 00 01 ba hex...). The mimetype video/mpeg can refer to a number of other formats, but only the described variants need to be supported for video. Other audio and

video formats can be supported depending on the abilities of the recognizers, and should be mentioned in the mimetypes attribute in the metadata.

**Auxiliary** files can be in any recognizer specific format. For example, a recognizer could produce a PNG image of the histogram distribution of some aspect of the input data, or some binary data file which can then be used as input by another recognizer from the same toolkit. Auxiliary files should only be used if tier, timeseries, audio or video files cannot reasonably be used to store the same data in open formats.

In the special case where many auxiliary files are used, a recognizer should declare usage of one file in the metadata and then derive the other filenames in some documented way. Note that this can create problems with portability and similar.

Input files can also be marked as optional. Such files could be tiers or even media used to guide the work of the recognizer by selecting areas of interest or by giving examples of an interesting piece of input or a desired result of the analysis for all or part of the input media. Send an empty file name for optional items that should be skipped.

The example *recognizer.cmdi* file describes a recognizer which needs one audio file as input and does not process other input files apart from that. The input file name is passed as a parameter called "source".

### 3.1.7 Output data

A recognizer can generate one or more files from the same set of file formats as used for input files. However, recognizers will usually only produce one or more timeseries and/or tier file and no audio or media files. The caller sends a target file name as input "parameter". The recognizer can fail to produce an output file for any reason, which is easy to detect for the caller. The caller should provide feedback about this to the user, in particular if the caller is interactive software.

Note that output files can also be marked as being optional in the metadata file: For such files, the caller does not have to provide a file name. If the recognizer receives an empty name for an optional file, it will not write the corresponding data to disk. Note that the item name still has to be sent, but will be empty. Of course recognizers can create temporary files using a system temp directory whenever this is necessary for them.

Output files can have their mimetypes specified in the metadata file. That way, the caller can know which types of result file formats to expect. XML and CSV timeseries and tier files have to be exactly in the formats described in this document, but auxiliary, audio and video files can use a variety of file formats. If possible, the metadata should list exactly one mimetype for each output file.

The example *recognizer.cmdi* file describes a recognizer which produces several tier files and optionally also a CSV timeseries file. One of the tier files created by the example recognizer is the "vowel_f0" tier. As described above, a tier is stored as an XML file with end time and one or more content values given for each item / timespan. It is up to the caller to interpret or display the content values. It can also just store the file, or use it as input for another recognizer.

### 3.1.8 Invocation *[Step 4]*

After a script or the user has provided all necessary input parameter values and set all necessary input file names and generated all necessary output file names, all "parameters" including input and output file names are sent to the recognizer via the standard input of the recognizer, encoded as XML. Note that unused file name items for unused input or output files are also sent, but as empty strings:

An extra parameter needed to store meta-data about individual invocation of a specific recognizer. The parameter XML block contains a special "InvocationContext" value. This param helps to track/log/archive invocations. This lets you archive the parameters that you use for a certain recognizer run, which is good for documenting what you did and good if you want to try again with a modified parameter set later. For ELAN, this means that it does not omit anything and for documentation and ELAN and recognizers it means that InvocationContext should be set (by ELAN) and ignored (by recognizers) respectively and is a reserved/special param name. The InvocationContext simply is the name of the recognizer (as specified in the CMDI) followed by a space and the RFC3339 timestamp when the parameter block was created. Recognizers can ignore this parameter, although old recognizers might log that they received a parameter that they had not asked for in the CMDI.

```
<?xml version="1.0" encoding="UTF-8"?>
<PARAM xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="file:avatech-call.xsd">
<param name=”InvocationContext”>TagVowels 2012-02-07 15:50:00+01:00</param>
<param name="max_f0">350</param>
  <param name="epsilon">1.0</param>
  <param name="vol_threshold">0.45</param>
  <param name="loglevel">verbose</param>
  <param name="source">/home/user/example/myspeech.wav</param>
  <param name="pitch_out"></param>
  <param name="vowel_f0">/home/user/example/myspeech_1.xml</param>
  <param name="vowel_db">/home/user/example/myspeech_2.xml</param>
  <param name="vowel_f0_initial">/home/user/example/myspeech_3.xml</param>
  <param name="vowel_f0_final">/home/user/example/myspeech_4.xml</param>
  <param name="vowel_f0_min">/home/user/example/myspeech_5.xml</param>
  <param name="vowel_f0_max">/home/user/example/myspeech_6.xml</param>
</PARAM>
```

### 3.1.9 Processing starts *[Step 5]*

As soon as the recognizer has received the parameters, the input files are opened and the processing starts. Any status and progress messages can be sent to the standard output of the recognizer process. Interactive software such as ELAN could display this output for the user, a script tool could send it to a log file. For more information about the input file formats, see section (
3.1.6 Input data).

### 3.1.10 Feedback *[Step 6]*

It is good to display logs from the recognizer immediately as they come in, so the user is warned early about problems and can consider parameter adjustments or consider aborting the computation.

Text lines written to standard output of the recognizer should start with a tag, "DEBUG:", "INFO:", "WARN:", "ERROR:", "PROGRESS:" or "RESULT:" to allow ELAN and other software to display information in appropriate ways. Tag and actual text content are separated by whitespace. If a recognizer is unable to provide tags, all content can be treated as "INFO:", which is suitable for display in a status bar or progress dialog. Lines with the "RESULT:" tag can be highlighted as a summary of the processing results; they can for example give some statistical information.

The "ERROR:" tag should only be used for problems which interrupt normal processing, while "WARN:" is to warn about problems which at most reduce result quality. The last item sent by the recognizer should be a line containing either the exact string **"RESULT: DONE."** or **"RESULT: FAILED."** followed by a last line break. If the recognizer exits or closes the connection without sending any of those two final lines, the caller should warn the user (or log) that the recognizer probably has crashed. It should not wait infinitely for a done or failed result in that case.

It is good to be able to communicate how far you are with processing in a marked way, in particular because ELAN is not showing you any logs while any recognizer runs. Instead, you have to wait until it either finishes or aborts. Giving a special syntax of INFO line a special status will allow ELAN to display information from such lines directly in the progress bar (as label) without having to wait until you can read the full "report". All of the following four variants of progress logging syntax are valid:

```
INFO: PROGRESS: 42.5% optional message
INFO: PROGRESS: 0.42 optional message
PROGRESS: 42.5% optional message
PROGRESS: 0.42 optional message
```

Progress messages should always specify an amount, either as a value between 0 and 1 or as a percentage. However, callers should gracefully proceed to show only the message if no percentage is given in the first word after “PROGRESS:” Also, old software can simply treat progress messages from recognizers as generic info messages.

The example recognizer could for example give some progress and status output similar to the one below (on standard output) if the "loglevel" parameter is set to "verbose":

```
INFO: Read 3158 data points, 1221 without pitch
DEBUG: Peak: 0.10..0.13 f0: 339.79 (327.58..345.50 L: 327.58 R: 345.50) dB: 78.15
DEBUG: Peak: 0.20..0.32 f0: 248.02 (217.19..316.64 L: 220.48 R: 311.79) dB: 73.78
...
... (some output omitted)
...
RESULT: Checked: 31570 msec in 3158 steps 61.3% voiced: n=65 avg=297.5 SD=286.2
RESULT:  4.7% mixed: n=104 avg=14.1 SD=36.7 8.6% line: n=99 avg=27.4 SD=39.0
RESULT:  25.0% peaks: n=125 avg=63.0 SD=51.8 8.4% dips: n=134 avg=19.9 SD=22.3
RESULT:  32.6% peak+: n=134 avg=76.9 SD=57.7 15.5% dip+: n=161 avg=30.4 SD=41.0
RESULT:  Hits: 32.6% t_hit: n=134 avg=76.9 SD=57.7 f0: n=134 avg=148.6 SD=47.9
RESULT: DONE.
```

On the other hand, it could also give much less verbose feedback, maybe selected by setting "loglevel" to "normal". Software which calls recognizers can display the incoming log in real time, possibly with syntax highlighting (ERROR: lines in red etc.) and it could save the log to a file. Only selected cases should require user interaction, for example

there could be a pop-up when recognition completes or fails, showing only the ERROR: and RESULT: lines. Note that flaws in recognizers could always cause non-standard situations, such as not producing any RESULT: line when a recognizer crashes, or stack traces being logged in the output without any keyword such as ERROR: being used to tag them. The caller must have sufficiently robust handling of logs and recognizer invocation. For example non-tagged log lines could be shown with the same highlighting as error lines and crashes could trigger a warning to the user.

### 3.1.11 Results stored *[Step 7]*

When calculations are finished, the recognizer process exits and the caller can open the result files, after checking whether and which of them were generated. When an error occurs, one or more of the output files can be missing. The status and progress messages should then give the user some detail information about the encountered problems.

See the section about output files for more information about the file formats used to store result data.

### 3.1.12 Installation of a self-standing recognizer component

A stand-alone recognizer can be installed locally or, by any system manager, on a dedicated server. It is agreed that such recognizers need to have a metadata description which will be stored at an agreed location, so that ELAN and ABAX can access it. The system manager needs to take care that the metadata includes binding information, i.e. the ways the recognizer is being invoked.

## 3.2 Plug-In Interface Specifications

ELAN 3.6.0 introduced the first version of a Recognizer API that allows for extending ELAN with pattern recognition based plug-in recognizer components. This interface provides a higher level of integration into the "caller" (i.e. ELAN) than the XML file based interfacing mechanisms described in this document do.

### 3.2.0 Changes made for ELAN 4.5.0

The composition of the graphical user interface of the recognizer panels has changed. The Selection panel that allowed adding multiple manual selections and/or complete tiers has been removed from the main panel - likewise for the list of media files. These items have been moved to the configuration or parameter panel of the recognizer. A new panel has been developed that allows to either add custom selections, or to select a tier, or to select a file as input to the recognizer. If a recognizer does not have its own control panel this new Selections panel will be added to the UI for the proper parameters based on the metadata. If the recognizer does have its own control panel, it can obtain the new Selections panel from the recognizer Host. The host will call the setMedia(List) method to pass the linked media files to the control panel.

The method getExampleSupport() has been removed from the Recognizer interface.

It is now possible to have multiple cmdi files in the same folder in the extensions folder, the name recognizer.cmdi is no longer mandatory.

### 3.2.1 Installation

Installation of recognizer plug-ins is done much in the same way as described above (in par. *3.1.12*). All necessary files need to be copied into a directory in the "extensions" folder inside ELAN's install folder. Initially the plug-in software needed to be contained in a (single) Java jar file, but this requirement has been extended by the option to have all necessary resources in a folder within the extensions folder. A *cmdi* file needs to be found in the root of the plug-in folder and all the Java and native libraries should be in that same folder or subfolders thereof. If the recognizer is packaged as a single jar file, the *cmdi* has to be in the root of that jar file.

### 3.2.2 Discovery of plug-ins

There are two ways in which installed plug-ins are discovered and loaded by ELAN. In both cases the *cmdi* file is searched for information concerning configurable parameters and for the class that implements the Recognizer interface. The **runLinux**, **runMac** and **runWin** attributes should contain this fully qualified class name. These attributes are also used to determine which platforms are supported, even if the recognizer is pure Java.

If the recognizer is packaged into a single jar file, it should either be fully self-contained or it should be able to load additional libraries independently. If the recognizer is installed in a separate folder within the extensions folder, the loading of both Java and native libraries is taken care of by ELAN. All Java and native libraries in the plug-in folder and subfolders are considered to be necessary. If feasible, an adaptor solution will be developed that will lift the burden of writing Java code for recognizers that are written in C or C++.

### 3.2.3 Parameter configuration

A recognizer plug-in may provide its own user interface component for configuring parameters and it must specify the parameters in the *recognizer.cmdi* file. In the latter case ELAN can instantiate a custom-made user interface component (based on the parameter specifications from the metadata) that allows the user to set parameter values.

### 3.2.4 The Recognizer API

The two main interfaces of the Recognizer API, which defines the communication between ELAN and a recognizer plug-in, are **Recognizer** and **RecognizerHost**. Any recognizer designed as a plug-in is required to implement the Recognizer interface. ELAN contains an implementation of RecognizerHost and other applications that aim to interact with a recognizer plug-in can do the same.

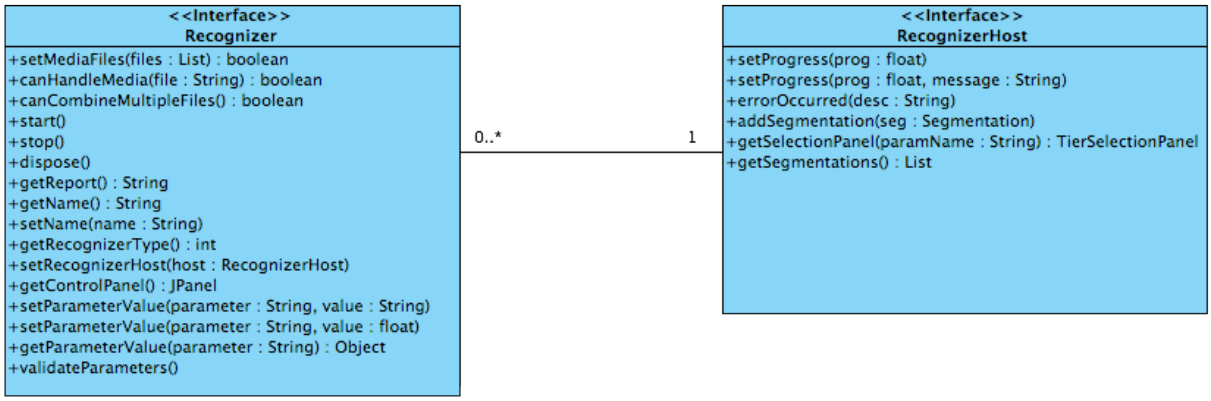



**Figure 4: Simplified diagram of Recognizer and RecognizerHost**

The flow of control after selection of a recognizer is as follows:

- The host calls setRecognizerHost to setup a communication channel between host and recognizer.

- The host iterates over the media files that are linked to the ELAN document and calls the canHandleMedia method of the selected recognizer. This step is merely intended to be able to disable in the user interface those media files that are not supported. The recognizer can do a simple check whether it supports the file or not. If it is possible to exclude certain files based on the metadata they will not be handed to the recognizer at all. The user can select which file(s) he/she wishes to hand over to the recognizer. The host calls setMediaFiles with the supported file(s).

- The host calls *getControlPanel* to see if the recognizer provides its own user interface component for setting parameters. If not, ELAN will build a user interface based on the *numparam*, *textparam, input and output* elements in the *cmdi* file.

- The host calls *getControlPanel* to see if the recognizer provides its own user interface component for setting parameters. If not, ELAN will build a user interface based on the *numparam*, *textparam, input and output* elements in the *cmdi* file.

- If the recognizer does have its own control panel and supports selections or tiers as input, it can obtain the tier selection panel by calling the getSelectionPanel method on the host. The recognizer can retrieve the selections made by calling the *getSelections* method on the TierSelectionPanel

- When the detection process is started there are two execution scenarios: if the recognizer does not provide a parameter panel but does specify configurable parameters in the metadata file, the host collects the user settings from the custom-made user interface and passes them to the recognizer by recursively calling one of the *setParameterValue* methods. If the recognizer does provide its own user interface for configurable settings, it is responsible for getting the settings in time. The host starts the detection process by calling the *start* method of the recognizer.

- Before a recognizer's start() method is called, the host will call the validateParameters method on the recognizer if it provides its own control panel to verify whether all parameters required to run the recognizer are valid.

- During execution the recognizer can send progress information to the host by calling *setProgress*. The value should be between 0 and 1; the value 1 indicates completion of the detection process.

- If an error occurs during execution the recognizer should notify the host via *errorOccurred.* The process is considered being terminated. The user can interrupt the detection process during execution by calling the *stop* method of the recognizer.

- Before finishing execution the recognizer should transfer any created segmentations to the host by calling *addSegmentation*.

- The user can decide to display a report on the process and the results by calling *getReport*. A recognizer is not required to return a meaningful overview.

- Segmentations created by the recognizer can be converted to tiers; ELAN calls the host's *getSegmentations* method for that purpose. Like any other tier, these tiers can be edited and can serve as input for a successive execution of the recognizer.

Apart from these interfaces the API specifies some data object classes forming several specializations of a time interval (segments) and a container for these segments

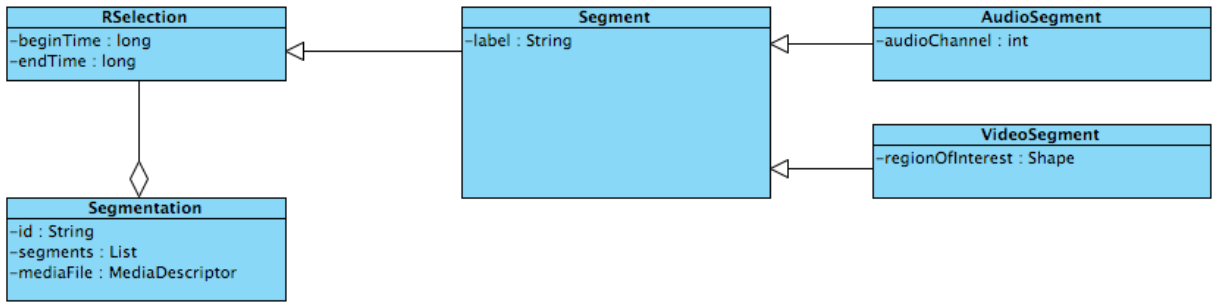



**Figure 5: Recognizer Data classes**

# Appendix I - XSD description of AVATecH file formats

The following five XSDs gives a "formal" description of the AVATecH file formats.

### avatech-call.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <xs:element name="PARAM">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="param"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="param">
    <xs:complexType mixed="true">
      <xs:attribute name="name" use="required" type="xs:NCName"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

### avatech-tier.xsd – classic format

(only one set of tiers, sharing one time segmentation)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <xs:element name="TIER">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="span"/>
      </xs:sequence>
      <xs:attribute name="columns" use="required" type="xs:string"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="span">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="v"/>
      </xs:sequence>
      <xs:attribute name="end" use="required" type="xs:decimal"/>
      <xs:attribute name="start" use="required" type="xs:decimal"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="v" type="xs:string"/>
</xs:schema>
```

### avatech-tiers.xsd - multitier format

(multiple sets of tiers, with independent time segmentation)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <xs:element name="TIERS">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="1" maxOccurs="unbounded" ref="TIER"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="TIER">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="span"/>
      </xs:sequence>
      <xs:attribute name="columns" use="required" type="xs:string"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="span">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="v"/>
      </xs:sequence>
      <xs:attribute name="end" use="required" type="xs:decimal"/>
      <xs:attribute name="start" use="required" type="xs:decimal"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="v" type="xs:string"/>
</xs:schema>
```

### avatech-timeseries.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <xs:element name="TIMESERIES">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="i"/>
      </xs:sequence>
      <xs:attribute name="columns" use="required" type="xs:string"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="i">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="v"/>
      </xs:sequence>
      <xs:attribute name="t" use="required" type="xs:decimal"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="v" type="xs:string"/>
</xs:schema>
```

**avatech-recognizer.xsd**

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:cmd="http://www.clarin.eu/cmd/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:dcr="http://www.isocat.org/ns/dcr" xmlns:ann="http://www.clarin.eu"
targetNamespace="http://www.clarin.eu/cmd/"
elementFormDefault="qualified">
  <xs:annotation>
    <xs:appinfo>
      <ann:Header>
        <ann:ID>clarin.eu:cr1:p_1393514855416</ann:ID>
        <ann:Name>RECOGNIZER</ann:Name>
        <ann:Description>See: http://www.mpi.nl/avatech/ and
http://tla.mpi.nl/projects_info/avatech/ for documentation.
Services are located at
http://catalog.clarin.eu/avatech/</ann:Description>
      </ann:Header>
    </xs:appinfo>
  </xs:annotation>
  <xs:import namespace="http://www.w3.org/XML/1998/namespace"
schemaLocation="http://www.w3.org/2001/xml.xsd"/>
  <xs:simpleType name="Resourcetype_simple">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Metadata">
        <xs:annotation>
          <xs:documentation>The ResourceProxy refers to another component
metadata
                            instance (e.g. for grouping metadata
descriptions into
                            collections)</xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="Resource">
        <xs:annotation>
          <xs:documentation>The ResourceProxy refers to a file that is
not a metadata
                            instance (e.g. a text
document)</xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="SearchService">
        <xs:annotation>
          <xs:documentation>The ResourceProxy refers to a (SRU/CQL) web
service that can be used to query the resource described in this
file</xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="SearchPage">
        <xs:annotation>
          <xs:documentation>The ResourceProxy refers to a web page that
can be used to query the resource described in this
file</xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="LandingPage">
        <xs:annotation>
          <xs:documentation>The ResourceProxy refers to a web page that
contains the "original context" of the resource described in this file
(e.g. repository web page displaying the metadata).</xs:documentation>
        </xs:annotation>
```

```
      </xs:enumeration>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="CMD">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Header">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="MdCreator" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
              <xs:element name="MdCreationDate" type="xs:date"
minOccurs="0"/>
              <xs:element name="MdSelfLink" type="xs:anyURI"
minOccurs="0"/>
              <xs:element name="MdProfile" type="xs:anyURI"
minOccurs="0"/>
              <xs:element name="MdCollectionDisplayName" type="xs:string"
minOccurs="0"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Resources">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="ResourceProxyList">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element maxOccurs="unbounded" minOccurs="0"
name="ResourceProxy">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element maxOccurs="1" minOccurs="1"
name="ResourceType">
                            <xs:complexType>
                              <xs:simpleContent>
                                <xs:extension
base="cmd:Resourcetype_simple">
                                  <xs:attribute name="mimetype"
type="xs:string"/>
                                </xs:extension>
                              </xs:simpleContent>
                            </xs:complexType>
                          </xs:element>
                          <xs:element maxOccurs="1" minOccurs="1"
name="ResourceRef" type="xs:anyURI"/>
                        </xs:sequence>
                        <xs:attribute name="id" type="xs:ID"
use="required"/>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="JournalFileProxyList">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element maxOccurs="unbounded" minOccurs="0"
name="JournalFileProxy">
                      <xs:complexType>
```

```
                        <xs:sequence>
                          <xs:element maxOccurs="1" minOccurs="1"
name="JournalFileRef" type="xs:anyURI"/>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="ResourceRelationList">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element maxOccurs="unbounded" minOccurs="0"
name="ResourceRelation">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element maxOccurs="1" minOccurs="1"
name="RelationType"/>
                          <xs:element maxOccurs="1" minOccurs="1"
name="Res1">
                            <xs:complexType>
                              <xs:attribute name="ref" type="xs:IDREF"/>
                            </xs:complexType>
                          </xs:element>
                          <xs:element maxOccurs="1" minOccurs="1"
name="Res2">
                            <xs:complexType>
                              <xs:attribute name="ref" type="xs:IDREF"/>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element minOccurs="0" name="IsPartOfList">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element maxOccurs="unbounded" minOccurs="0"
name="IsPartOf" type="xs:anyURI"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Components">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="RECOGNIZER" minOccurs="1" maxOccurs="1">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="recognizer"
dcr:datcat="http://www.isocat.org/datcat/DC-4584" minOccurs="1"
maxOccurs="1" ann:documentation="description of avatech or auvis
recognizers and their command line execution arguments"
ann:displaypriority="5">
                      <xs:complexType>
                        <xs:simpleContent>
```

```
                        <xs:extension base="xs:string">
                          <xs:attribute name="recognizerType"
dcr:datcat="http://www.isocat.org/datcat/DC-3786">
                            <xs:simpleType>
                              <xs:restriction base="xs:string">
                                <xs:enumeration value="direct"
dcr:datcat="http://www.isocat.org/datcat/DC-4445" ann:label=""/>
                                <xs:enumeration value="local"
dcr:datcat="http://www.isocat.org/datcat/DC-4585" ann:label=""/>
                                <xs:enumeration value="shared"
dcr:datcat="http://www.isocat.org/datcat/DC-4583" ann:label=""/>
                                <xs:enumeration value="remote"
dcr:datcat="http://www.isocat.org/datcat/DC-4583" ann:label=""/>
                              </xs:restriction>
                            </xs:simpleType>
                          </xs:attribute>
                          <xs:attribute name="runLinux"
dcr:datcat="http://www.isocat.org/datcat/DC-2569" type="xs:string"/>
                          <xs:attribute name="runMac"
dcr:datcat="http://www.isocat.org/datcat/DC-2569" type="xs:string"/>
                          <xs:attribute name="runWin"
dcr:datcat="http://www.isocat.org/datcat/DC-2569" type="xs:string"/>
                          <xs:attribute name="info"
dcr:datcat="http://www.isocat.org/datcat/DC-2520" type="xs:string"/>
                        </xs:extension>
                      </xs:simpleContent>
                    </xs:complexType>
                  </xs:element>
                  <xs:element name="documentation"
dcr:datcat="http://www.isocat.org/datcat/DC-2656" type="xs:anyURI"
minOccurs="0" maxOccurs="1" ann:documentation="link to a document
describing the recognizer (version, license, homepage, usage...)"/>
                  <xs:element name="input"
dcr:datcat="http://www.isocat.org/datcat/DC-3815" minOccurs="0"
maxOccurs="unbounded" ann:documentation="description of some input file
used by recognizers">
                    <xs:complexType>
                      <xs:simpleContent>
                        <xs:extension base="xs:string">
                          <xs:attribute name="type"
dcr:datcat="http://www.isocat.org/datcat/DC-3799">
                            <xs:simpleType>
                              <xs:restriction base="xs:string">
                                <xs:enumeration value="audio"
dcr:datcat="" ann:label="audio file, wav or other"/>
                                <xs:enumeration value="video"
dcr:datcat="" ann:label="video file, mpeg or other"/>
                                <xs:enumeration value="tier"
dcr:datcat="" ann:label="annotation tier file, xml"/>
                                <xs:enumeration value="csvtier"
dcr:datcat="" ann:label="annotation tier file, csv"/>
                                <xs:enumeration value="timeseries"
dcr:datcat="" ann:label="timed information, e.g. numerical, xml"/>
                                <xs:enumeration value="csvtimeseries"
dcr:datcat="" ann:label="timed information, e.g. numerical, csv"/>
                                <xs:enumeration value="auxiliary"
dcr:datcat="" ann:label="any other file"/>
                                <xs:enumeration value="multitier"
dcr:datcat="" ann:label="multiple annotation tier file, xml"/>
                              </xs:restriction>
```

```
                              </xs:simpleType>
                            </xs:attribute>
                            <xs:attribute name="mimetypes"
dcr:datcat="http://www.isocat.org/datcat/DC-2571" type="xs:string"/>
                            <xs:attribute name="optional"
type="xs:boolean"/>
                            <xs:attribute name="info"
dcr:datcat="http://www.isocat.org/datcat/DC-2520" type="xs:string"/>
                            <xs:attribute name="level">
                              <xs:simpleType>
                                <xs:restriction base="xs:string">
                                  <xs:enumeration value="basic"
dcr:datcat="" ann:label="general setting"/>
                                  <xs:enumeration value="advanced"
dcr:datcat="" ann:label="setting specific to advanced users"/>
                                </xs:restriction>
                              </xs:simpleType>
                            </xs:attribute>
                          </xs:extension>
                        </xs:simpleContent>
                      </xs:complexType>
                    </xs:element>
                    <xs:element name="numparam"
dcr:datcat="http://www.isocat.org/datcat/DC-3825" minOccurs="0"
maxOccurs="unbounded" ann:documentation="definition of a numerical input
parameter for recognizers">
                      <xs:complexType>
                        <xs:simpleContent>
                          <xs:extension base="xs:string">
                            <xs:attribute name="min"
dcr:datcat="http://www.isocat.org/datcat/DC-1978" type="xs:float"/>
                            <xs:attribute name="max"
dcr:datcat="http://www.isocat.org/datcat/DC-1978" type="xs:float"/>
                            <xs:attribute name="default"
dcr:datcat="http://www.isocat.org/datcat/DC-1978" type="xs:float"/>
                            <xs:attribute name="info"
dcr:datcat="http://www.isocat.org/datcat/DC-2520" type="xs:string"/>
                            <xs:attribute name="level">
                              <xs:simpleType>
                                <xs:restriction base="xs:string">
                                  <xs:enumeration value="basic"
dcr:datcat="" ann:label="general setting"/>
                                  <xs:enumeration value="advanced"
dcr:datcat="" ann:label="setting specific to advanced users"/>
                                </xs:restriction>
                              </xs:simpleType>
                            </xs:attribute>
                            <xs:attribute name="type">
                              <xs:simpleType>
                                <xs:restriction base="xs:string">
                                  <xs:enumeration value="int"
dcr:datcat="" ann:label="integer value"/>
                                  <xs:enumeration value="float"
dcr:datcat="" ann:label="numerical 1.234 style value, or integer if no
fractional part needed"/>
                                </xs:restriction>
                              </xs:simpleType>
                            </xs:attribute>
                          </xs:extension>
                        </xs:simpleContent>
```

```
                    </xs:complexType>
                  </xs:element>
                  <xs:element name="textparam"
dcr:datcat="http://www.isocat.org/datcat/DC-3825" minOccurs="0"
maxOccurs="unbounded" ann:documentation="description of a textual or
choice parameter for recognizers">
                    <xs:complexType>
                      <xs:simpleContent>
                        <xs:extension base="xs:string">
                          <xs:attribute name="convoc"
dcr:datcat="http://www.isocat.org/datcat/DC-1978" type="xs:string"/>
                          <xs:attribute name="default"
dcr:datcat="http://www.isocat.org/datcat/DC-1978" type="xs:string"/>
                          <xs:attribute name="info"
dcr:datcat="http://www.isocat.org/datcat/DC-2520" type="xs:string"/>
                          <xs:attribute name="level">
                            <xs:simpleType>
                              <xs:restriction base="xs:string">
                                <xs:enumeration value="basic"
dcr:datcat="" ann:label="general setting"/>
                                <xs:enumeration value="advanced"
dcr:datcat="" ann:label="setting specific to advanced users"/>
                              </xs:restriction>
                            </xs:simpleType>
                          </xs:attribute>
                        </xs:extension>
                      </xs:simpleContent>
                    </xs:complexType>
                  </xs:element>
                  <xs:element name="output"
dcr:datcat="http://www.isocat.org/datcat/DC-3804" minOccurs="0"
maxOccurs="unbounded" ann:documentation="description of some output file
used by recognizers">
                    <xs:complexType>
                      <xs:simpleContent>
                        <xs:extension base="xs:string">
                          <xs:attribute name="type"
dcr:datcat="http://www.isocat.org/datcat/DC-1978">
                            <xs:simpleType>
                              <xs:restriction base="xs:string">
                                <xs:enumeration value="audio"
dcr:datcat="" ann:label="audio file, wav or other"/>
                                <xs:enumeration value="video"
dcr:datcat="" ann:label="video file, mpeg or other"/>
                                <xs:enumeration value="tier"
dcr:datcat="" ann:label="annotation tier file, xml"/>
                                <xs:enumeration value="csvtier"
dcr:datcat="" ann:label="annotation tier file, csv"/>
                                <xs:enumeration value="timeseries"
dcr:datcat="" ann:label="timed information file, e.g. numerical, xml"/>
                                <xs:enumeration value="csvtimeseries"
dcr:datcat="" ann:label="timed information, e.g numerical, csv"/>
                                <xs:enumeration value="auxiliary"
dcr:datcat="" ann:label="any other file"/>
                                <xs:enumeration value="multitier"
dcr:datcat="" ann:label="multiple annotation tier file, xml"/>
                              </xs:restriction>
                            </xs:simpleType>
                          </xs:attribute>
                          <xs:attribute name="mimetypes"
```

```
dcr:datcat="http://www.isocat.org/datcat/DC-2571" type="xs:string"/>
                            <xs:attribute name="optional"
type="xs:boolean"/>
                            <xs:attribute name="info"
dcr:datcat="http://www.isocat.org/datcat/DC-2520" type="xs:string"/>
                            <xs:attribute name="level">
                              <xs:simpleType>
                                <xs:restriction base="xs:string">
                                  <xs:enumeration value="basic"
dcr:datcat="" ann:label="general setting"/>
                                  <xs:enumeration value="advanced"
dcr:datcat="" ann:label="setting specific to advanced users"/>
                                </xs:restriction>
                              </xs:simpleType>
                            </xs:attribute>
                          </xs:extension>
                        </xs:simpleContent>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                  <xs:attribute name="ref" type="xs:IDREFS"/>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="CMDVersion" fixed="1.1" use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

# Appendix II - Recognizer metadata file format

## Appendix - Example CMDI file (*recognizer.cmdi*)

Thanks to Dieter van Uytvanck for help with the design of the recognizer metadata CMDI file format. The CMDI defines which user controllable options exist for a recognizer.

Note that the <Header> and <Resources> sections of CMDI are usually not used in AVATecH recognizer metadata files. They exist as part of the CMDI standard and can be left empty in recognizer metadata.

```
<?xml version="1.0" encoding="UTF-8"?>
<CMD CMDVersion="1.1" xmlns="http://www.clarin.eu/cmd/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.clarin.eu/cmd/
http://catalog.clarin.eu/ds/ComponentRegistry/rest/registry/profiles/clar
in.eu:cr1:p_1393514855416/xsd">
    <Header/>
    <Resources>
        <ResourceProxyList/>
        <JournalFileProxyList/>
        <ResourceRelationList/>
    </Resources>
    <Components>
        <RECOGNIZER>
            <recognizer recognizerType="local" runLinux="PitchPraat.sh"
runMac="PitchPraat.sh" runWin="PP.bat" info="Tag vowels (volume peaks of
voiced spans)">TagVowels</recognizer>
            <documentation>tagvowels.html</documentation>
            <input type="audio" optional="false" mimetypes="audio/x-wav"
info="Mono or stereo WAV to scan">source</input>
            <numparam type="int" level="basic" min="150" max="1500"
default="550" info="Pitch ceiling [Hz], eg. 300 male / 550
female">max_f0</numparam>
            <numparam min="0.5" max="5.0" default="2.0" info="Intensity
change [dB] to start/end a peak">epsilon</numparam>
            <numparam min="0.01" max="0.3" default="0.03" info="Min
amplitude (0..1) for pitch analysis">vol_threshold</numparam>
            <numparam min="0.2" max="0.8" default="0.45" info="Min
voicing of voiced timespans">voicedness_threshold</numparam>
            <textparam convoc="normal verbose debug" default="normal"
info="Adjust amount of messages during calculation
here">loglevel</textparam>
            <output type="csvtimeseries" optional="true"
mimetypes="text/csv" info="Pitch/undefined and volume, 100
samples/s">pitch_out</output>
            <output type="tier" optional="false" mimetypes="text/xml"
info="Vowel segments and their average f0 [Hz]">vowel_f0</output>
            <output type="tier" optional="false" mimetypes="text/xml"
info="Vowel segments and average intensity [db]">vowel_db</output>
            <output type="tier" optional="false" mimetypes="text/xml"
info="Vowel segments and initial f0 [Hz]">vowel_f0_initial</output>
            <output type="tier" optional="false" mimetypes="text/xml"
info="Vowel segments and their final f0 [Hz]">vowel_f0_final</output>
            <output type="tier" optional="false" mimetypes="text/xml"
info="Vowel segments and their minimum f0 [Hz]">vowel_f0_min</output>
```

```
            <output type="tier" optional="false" mimetypes="text/xml"
info="Vowel segments and their maximum f0 [Hz]">vowel_f0_max</output>
        </RECOGNIZER>
    </Components>
</CMD>
```

## Appendix - Recognizer CMDI definition

Note that the <Header> and <Resources> sections of CMDI are usually not filled in AVATecH recognizer metadata files. They exist as part of the CMDI standard and can be left empty in recognizer metadata.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CMD_ComponentSpec isProfile="true"
xsi:schemaLocation="http://www.clarin.eu/cmd
https://infra.clarin.eu/cmd/general-component-schema.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Header>
        <ID>clarin.eu:cr1:p_1393514855416</ID>
        <Name>RECOGNIZER</Name>
        <Description>See: http://www.mpi.nl/avatech/ and
http://tla.mpi.nl/projects_info/avatech/ for documentation.
Services are located at
http://catalog.clarin.eu/avatech/</Description>
    </Header>
    <CMD_Component name="RECOGNIZER" CardinalityMin="1"
CardinalityMax="1">
        <CMD_Element name="recognizer"
ConceptLink="http://www.isocat.org/datcat/DC-4584"
ValueScheme="string" CardinalityMin="1" CardinalityMax="1"
Documentation="description of avatech or auvis recognizers
and their command line execution arguments"
DisplayPriority="5" Multilingual="false">
            <AttributeList>
                <Attribute>
                    <Name>recognizerType</Name>

<ConceptLink>http://www.isocat.org/datcat/DC-
3786</ConceptLink>
                    <ValueScheme>
                        <enumeration>
                            <item
ConceptLink="http://www.isocat.org/datcat/DC-4445"
AppInfo="">direct</item>
                            <item
ConceptLink="http://www.isocat.org/datcat/DC-4585"
AppInfo="">local</item>
                            <item
ConceptLink="http://www.isocat.org/datcat/DC-4583"
AppInfo="">shared</item>
                            <item
ConceptLink="http://www.isocat.org/datcat/DC-4583"
AppInfo="">remote</item>
                        </enumeration>
                    </ValueScheme>
                </Attribute>
                <Attribute>
                    <Name>runLinux</Name>
```

```
<ConceptLink>http://www.isocat.org/datcat/DC-
2569</ConceptLink>
                    <Type>string</Type>
                </Attribute>
                <Attribute>
                    <Name>runMac</Name>

<ConceptLink>http://www.isocat.org/datcat/DC-
2569</ConceptLink>
                    <Type>string</Type>
                </Attribute>
                <Attribute>
                    <Name>runWin</Name>

<ConceptLink>http://www.isocat.org/datcat/DC-
2569</ConceptLink>
                    <Type>string</Type>
                </Attribute>
                <Attribute>
                    <Name>info</Name>

<ConceptLink>http://www.isocat.org/datcat/DC-
2520</ConceptLink>
                    <Type>string</Type>
                </Attribute>
            </AttributeList>
        </CMD_Element>
        <CMD_Element name="documentation"
ConceptLink="http://www.isocat.org/datcat/DC-2656"
ValueScheme="anyURI" CardinalityMin="0" CardinalityMax="1"
Documentation="link to a document describing the recognizer
(version, license, homepage, usage...)"/>
        <CMD_Element name="input"
ConceptLink="http://www.isocat.org/datcat/DC-3815"
ValueScheme="string" CardinalityMin="0"
CardinalityMax="unbounded" Documentation="description of
some input file used by recognizers" Multilingual="false">
            <AttributeList>
                <Attribute>
                    <Name>type</Name>

<ConceptLink>http://www.isocat.org/datcat/DC-
3799</ConceptLink>
                    <ValueScheme>
                        <enumeration>
                            <item ConceptLink=""
AppInfo="audio file, wav or other">audio</item>
                            <item ConceptLink=""
AppInfo="video file, mpeg or other">video</item>
                            <item ConceptLink=""
AppInfo="annotation tier file, xml">tier</item>
                            <item ConceptLink=""
AppInfo="annotation tier file, csv">csvtier</item>
                            <item ConceptLink=""
AppInfo="timed information, e.g. numerical,
xml">timeseries</item>
                            <item ConceptLink=""
AppInfo="timed information, e.g. numerical,
csv">csvtimeseries</item>
```

```
                            <item ConceptLink=""
AppInfo="any other file">auxiliary</item>
                            <item ConceptLink=""
AppInfo="multiple annotation tier file,
xml">multitier</item>
                        </enumeration>
                    </ValueScheme>
                </Attribute>
                <Attribute>
                    <Name>mimetypes</Name>

<ConceptLink>http://www.isocat.org/datcat/DC-
2571</ConceptLink>
                    <Type>string</Type>
                </Attribute>
                <Attribute>
                    <Name>optional</Name>
                    <Type>boolean</Type>
                </Attribute>
                <Attribute>
                    <Name>info</Name>

<ConceptLink>http://www.isocat.org/datcat/DC-
2520</ConceptLink>
                    <Type>string</Type>
                </Attribute>
                <Attribute>
                    <Name>level</Name>
                    <ValueScheme>
                        <enumeration>
                            <item ConceptLink=""
AppInfo="general setting">basic</item>
                            <item ConceptLink=""
AppInfo="setting specific to advanced users">advanced</item>
                        </enumeration>
                    </ValueScheme>
                </Attribute>
            </AttributeList>
        </CMD_Element>
        <CMD_Element name="numparam"
ConceptLink="http://www.isocat.org/datcat/DC-3825"
ValueScheme="string" CardinalityMin="0"
CardinalityMax="unbounded" Documentation="definition of a
numerical input parameter for recognizers"
Multilingual="false">
            <AttributeList>
                <Attribute>
                    <Name>min</Name>

<ConceptLink>http://www.isocat.org/datcat/DC-
1978</ConceptLink>
                    <Type>float</Type>
                </Attribute>
                <Attribute>
                    <Name>max</Name>

<ConceptLink>http://www.isocat.org/datcat/DC-
1978</ConceptLink>
                    <Type>float</Type>
                </Attribute>
```

```
                <Attribute>
                    <Name>default</Name>

<ConceptLink>http://www.isocat.org/datcat/DC-
1978</ConceptLink>
                    <Type>float</Type>
                </Attribute>
                <Attribute>
                    <Name>info</Name>

<ConceptLink>http://www.isocat.org/datcat/DC-
2520</ConceptLink>
                    <Type>string</Type>
                </Attribute>
                <Attribute>
                    <Name>level</Name>
                    <ValueScheme>
                        <enumeration>
                            <item ConceptLink=""
AppInfo="general setting">basic</item>
                            <item ConceptLink=""
AppInfo="setting specific to advanced users">advanced</item>
                        </enumeration>
                    </ValueScheme>
                </Attribute>
                <Attribute>
                    <Name>type</Name>
                    <ValueScheme>
                        <enumeration>
                            <item ConceptLink=""
AppInfo="integer value">int</item>
                            <item ConceptLink=""
AppInfo="numerical 1.234 style value, or integer if no
fractional part needed">float</item>
                        </enumeration>
                    </ValueScheme>
                </Attribute>
            </AttributeList>
        </CMD_Element>
        <CMD_Element name="textparam"
ConceptLink="http://www.isocat.org/datcat/DC-3825"
ValueScheme="string" CardinalityMin="0"
CardinalityMax="unbounded" Documentation="description of a
textual or choice parameter for recognizers"
Multilingual="false">
            <AttributeList>
                <Attribute>
                    <Name>convoc</Name>

<ConceptLink>http://www.isocat.org/datcat/DC-
1978</ConceptLink>
                    <Type>string</Type>
                </Attribute>
                <Attribute>
                    <Name>default</Name>

<ConceptLink>http://www.isocat.org/datcat/DC-
1978</ConceptLink>
                    <Type>string</Type>
                </Attribute>
```

```
                <Attribute>
                    <Name>info</Name>

<ConceptLink>http://www.isocat.org/datcat/DC-
2520</ConceptLink>
                    <Type>string</Type>
                </Attribute>
                <Attribute>
                    <Name>level</Name>
                    <ValueScheme>
                        <enumeration>
                            <item ConceptLink=""
AppInfo="general setting">basic</item>
                            <item ConceptLink=""
AppInfo="setting specific to advanced users">advanced</item>
                        </enumeration>
                    </ValueScheme>
                </Attribute>
            </AttributeList>
        </CMD_Element>
        <CMD_Element name="output"
ConceptLink="http://www.isocat.org/datcat/DC-3804"
ValueScheme="string" CardinalityMin="0"
CardinalityMax="unbounded" Documentation="description of
some output file used by recognizers" Multilingual="false">
            <AttributeList>
                <Attribute>
                    <Name>type</Name>

<ConceptLink>http://www.isocat.org/datcat/DC-
1978</ConceptLink>
                    <ValueScheme>
                        <enumeration>
                            <item ConceptLink=""
AppInfo="audio file, wav or other">audio</item>
                            <item ConceptLink=""
AppInfo="video file, mpeg or other">video</item>
                            <item ConceptLink=""
AppInfo="annotation tier file, xml">tier</item>
                            <item ConceptLink=""
AppInfo="annotation tier file, csv">csvtier</item>
                            <item ConceptLink=""
AppInfo="timed information file, e.g. numerical,
xml">timeseries</item>
                            <item ConceptLink=""
AppInfo="timed information, e.g numerical,
csv">csvtimeseries</item>
                            <item ConceptLink=""
AppInfo="any other file">auxiliary</item>
                            <item ConceptLink=""
AppInfo="multiple annotation tier file,
xml">multitier</item>
                        </enumeration>
                    </ValueScheme>
                </Attribute>
                <Attribute>
                    <Name>mimetypes</Name>

<ConceptLink>http://www.isocat.org/datcat/DC-
2571</ConceptLink>
```

```
                    <Type>string</Type>
                </Attribute>
                <Attribute>
                    <Name>optional</Name>
                    <Type>boolean</Type>
                </Attribute>
                <Attribute>
                    <Name>info</Name>

<ConceptLink>http://www.isocat.org/datcat/DC-
2520</ConceptLink>
                    <Type>string</Type>
                </Attribute>
                <Attribute>
                    <Name>level</Name>
                    <ValueScheme>
                        <enumeration>
                            <item ConceptLink=""
AppInfo="general setting">basic</item>
                            <item ConceptLink=""
AppInfo="setting specific to advanced users">advanced</item>
                        </enumeration>
                    </ValueScheme>
                </Attribute>
            </AttributeList>
        </CMD_Element>
    </CMD_Component>
</CMD_ComponentSpec>
```

## Appendix - Recognizer CMDI XSD (avatech-recognizer.xsd)

Note that the <Header> and <Resources> sections of CMDI are usually not filled in AVATecH recognizer metadata files. They exist as part of the CMDI standard and can be left empty in recognizer metadata.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:cmd="http://www.clarin.eu/cmd/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:dcr="http://www.isocat.org/ns/dcr"
xmlns:ann="http://www.clarin.eu"
targetNamespace="http://www.clarin.eu/cmd/"
elementFormDefault="qualified">
  <xs:annotation>
    <xs:appinfo>
      <ann:Header>
        <ann:ID>clarin.eu:cr1:p_1393514855416</ann:ID>
        <ann:Name>RECOGNIZER</ann:Name>
        <ann:Description>See: http://www.mpi.nl/avatech/ and
http://tla.mpi.nl/projects_info/avatech/ for documentation.
Services are located at
http://catalog.clarin.eu/avatech/</ann:Description>
      </ann:Header>
    </xs:appinfo>
  </xs:annotation>
  <xs:import namespace="http://www.w3.org/XML/1998/namespace"
schemaLocation="http://www.w3.org/2001/xml.xsd"/>
  <xs:simpleType name="Resourcetype_simple">
```

```
    <xs:restriction base="xs:string">
      <xs:enumeration value="Metadata">
        <xs:annotation>
          <xs:documentation>The ResourceProxy refers to another
component metadata
                            instance (e.g. for grouping metadata
descriptions into
                            collections)</xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="Resource">
        <xs:annotation>
          <xs:documentation>The ResourceProxy refers to a file that is
not a metadata
                            instance (e.g. a text
document)</xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="SearchService">
        <xs:annotation>
          <xs:documentation>The ResourceProxy refers to a (SRU/CQL) web
service that can be used to query the resource described in this
file</xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="SearchPage">
        <xs:annotation>
          <xs:documentation>The ResourceProxy refers to a web page that
can be used to query the resource described in this
file</xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="LandingPage">
        <xs:annotation>
          <xs:documentation>The ResourceProxy refers to a web page that
contains the "original context" of the resource described in this file
(e.g. repository web page displaying the metadata).</xs:documentation>
        </xs:annotation>
      </xs:enumeration>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="CMD">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Header">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="MdCreator" type="xs:string"
minOccurs="0" maxOccurs="unbounded"/>
              <xs:element name="MdCreationDate" type="xs:date"
minOccurs="0"/>
              <xs:element name="MdSelfLink" type="xs:anyURI"
minOccurs="0"/>
              <xs:element name="MdProfile" type="xs:anyURI"
minOccurs="0"/>
              <xs:element name="MdCollectionDisplayName"
type="xs:string" minOccurs="0"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
```

```
        <xs:element name="Resources">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="ResourceProxyList">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element maxOccurs="unbounded" minOccurs="0"
name="ResourceProxy">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element maxOccurs="1" minOccurs="1"
name="ResourceType">
                            <xs:complexType>
                              <xs:simpleContent>
                                <xs:extension
base="cmd:Resourcetype_simple">
                                  <xs:attribute name="mimetype"
type="xs:string"/>
                                </xs:extension>
                              </xs:simpleContent>
                            </xs:complexType>
                          </xs:element>
                          <xs:element maxOccurs="1" minOccurs="1"
name="ResourceRef" type="xs:anyURI"/>
                        </xs:sequence>
                        <xs:attribute name="id" type="xs:ID"
use="required"/>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="JournalFileProxyList">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element maxOccurs="unbounded" minOccurs="0"
name="JournalFileProxy">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element maxOccurs="1" minOccurs="1"
name="JournalFileRef" type="xs:anyURI"/>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="ResourceRelationList">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element maxOccurs="unbounded" minOccurs="0"
name="ResourceRelation">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element maxOccurs="1" minOccurs="1"
name="RelationType"/>
                          <xs:element maxOccurs="1" minOccurs="1"
name="Res1">
                            <xs:complexType>
                              <xs:attribute name="ref" type="xs:IDREF"/>
```

```
                            </xs:complexType>
                          </xs:element>
                          <xs:element maxOccurs="1" minOccurs="1"
name="Res2">
                            <xs:complexType>
                              <xs:attribute name="ref" type="xs:IDREF"/>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element minOccurs="0" name="IsPartOfList">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element maxOccurs="unbounded" minOccurs="0"
name="IsPartOf" type="xs:anyURI"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Components">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="RECOGNIZER" minOccurs="1" maxOccurs="1">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="recognizer"
dcr:datcat="http://www.isocat.org/datcat/DC-4584" minOccurs="1"
maxOccurs="1" ann:documentation="description of avatech or auvis
recognizers and their command line execution arguments"
ann:displaypriority="5">
                      <xs:complexType>
                        <xs:simpleContent>
                          <xs:extension base="xs:string">
                            <xs:attribute name="recognizerType"
dcr:datcat="http://www.isocat.org/datcat/DC-3786">
                              <xs:simpleType>
                                <xs:restriction base="xs:string">
                                  <xs:enumeration value="direct"
dcr:datcat="http://www.isocat.org/datcat/DC-4445" ann:label=""/>
                                  <xs:enumeration value="local"
dcr:datcat="http://www.isocat.org/datcat/DC-4585" ann:label=""/>
                                  <xs:enumeration value="shared"
dcr:datcat="http://www.isocat.org/datcat/DC-4583" ann:label=""/>
                                  <xs:enumeration value="remote"
dcr:datcat="http://www.isocat.org/datcat/DC-4583" ann:label=""/>
                                </xs:restriction>
                              </xs:simpleType>
                            </xs:attribute>
                            <xs:attribute name="runLinux"
dcr:datcat="http://www.isocat.org/datcat/DC-2569" type="xs:string"/>
                            <xs:attribute name="runMac"
dcr:datcat="http://www.isocat.org/datcat/DC-2569" type="xs:string"/>
                            <xs:attribute name="runWin"
dcr:datcat="http://www.isocat.org/datcat/DC-2569" type="xs:string"/>
```

```
                            <xs:attribute name="info"
dcr:datcat="http://www.isocat.org/datcat/DC-2520" type="xs:string"/>
                          </xs:extension>
                        </xs:simpleContent>
                      </xs:complexType>
                    </xs:element>
                    <xs:element name="documentation"
dcr:datcat="http://www.isocat.org/datcat/DC-2656" type="xs:anyURI"
minOccurs="0" maxOccurs="1" ann:documentation="link to a document
describing the recognizer (version, license, homepage, usage...)"/>
                    <xs:element name="input"
dcr:datcat="http://www.isocat.org/datcat/DC-3815" minOccurs="0"
maxOccurs="unbounded" ann:documentation="description of some input file
used by recognizers">
                      <xs:complexType>
                        <xs:simpleContent>
                          <xs:extension base="xs:string">
                            <xs:attribute name="type"
dcr:datcat="http://www.isocat.org/datcat/DC-3799">
                              <xs:simpleType>
                                <xs:restriction base="xs:string">
                                  <xs:enumeration value="audio"
dcr:datcat="" ann:label="audio file, wav or other"/>
                                  <xs:enumeration value="video"
dcr:datcat="" ann:label="video file, mpeg or other"/>
                                  <xs:enumeration value="tier"
dcr:datcat="" ann:label="annotation tier file, xml"/>
                                  <xs:enumeration value="csvtier"
dcr:datcat="" ann:label="annotation tier file, csv"/>
                                  <xs:enumeration value="timeseries"
dcr:datcat="" ann:label="timed information, e.g. numerical, xml"/>
                                  <xs:enumeration value="csvtimeseries"
dcr:datcat="" ann:label="timed information, e.g. numerical, csv"/>
                                  <xs:enumeration value="auxiliary"
dcr:datcat="" ann:label="any other file"/>
                                  <xs:enumeration value="multitier"
dcr:datcat="" ann:label="multiple annotation tier file, xml"/>
                                </xs:restriction>
                              </xs:simpleType>
                            </xs:attribute>
                            <xs:attribute name="mimetypes"
dcr:datcat="http://www.isocat.org/datcat/DC-2571" type="xs:string"/>
                            <xs:attribute name="optional"
type="xs:boolean"/>
                            <xs:attribute name="info"
dcr:datcat="http://www.isocat.org/datcat/DC-2520" type="xs:string"/>
                            <xs:attribute name="level">
                              <xs:simpleType>
                                <xs:restriction base="xs:string">
                                  <xs:enumeration value="basic"
dcr:datcat="" ann:label="general setting"/>
                                  <xs:enumeration value="advanced"
dcr:datcat="" ann:label="setting specific to advanced users"/>
                                </xs:restriction>
                              </xs:simpleType>
                            </xs:attribute>
                          </xs:extension>
                        </xs:simpleContent>
                      </xs:complexType>
                    </xs:element>
```

```
                    <xs:element name="numparam"
dcr:datcat="http://www.isocat.org/datcat/DC-3825" minOccurs="0"
maxOccurs="unbounded" ann:documentation="definition of a numerical input
parameter for recognizers">
                      <xs:complexType>
                        <xs:simpleContent>
                          <xs:extension base="xs:string">
                            <xs:attribute name="min"
dcr:datcat="http://www.isocat.org/datcat/DC-1978" type="xs:float"/>
                            <xs:attribute name="max"
dcr:datcat="http://www.isocat.org/datcat/DC-1978" type="xs:float"/>
                            <xs:attribute name="default"
dcr:datcat="http://www.isocat.org/datcat/DC-1978" type="xs:float"/>
                            <xs:attribute name="info"
dcr:datcat="http://www.isocat.org/datcat/DC-2520" type="xs:string"/>
                            <xs:attribute name="level">
                              <xs:simpleType>
                                <xs:restriction base="xs:string">
                                  <xs:enumeration value="basic"
dcr:datcat="" ann:label="general setting"/>
                                  <xs:enumeration value="advanced"
dcr:datcat="" ann:label="setting specific to advanced users"/>
                                </xs:restriction>
                              </xs:simpleType>
                            </xs:attribute>
                            <xs:attribute name="type">
                              <xs:simpleType>
                                <xs:restriction base="xs:string">
                                  <xs:enumeration value="int"
dcr:datcat="" ann:label="integer value"/>
                                  <xs:enumeration value="float"
dcr:datcat="" ann:label="numerical 1.234 style value, or integer if no
fractional part needed"/>
                                </xs:restriction>
                              </xs:simpleType>
                            </xs:attribute>
                          </xs:extension>
                        </xs:simpleContent>
                      </xs:complexType>
                    </xs:element>
                    <xs:element name="textparam"
dcr:datcat="http://www.isocat.org/datcat/DC-3825" minOccurs="0"
maxOccurs="unbounded" ann:documentation="description of a textual or
choice parameter for recognizers">
                      <xs:complexType>
                        <xs:simpleContent>
                          <xs:extension base="xs:string">
                            <xs:attribute name="convoc"
dcr:datcat="http://www.isocat.org/datcat/DC-1978" type="xs:string"/>
                            <xs:attribute name="default"
dcr:datcat="http://www.isocat.org/datcat/DC-1978" type="xs:string"/>
                            <xs:attribute name="info"
dcr:datcat="http://www.isocat.org/datcat/DC-2520" type="xs:string"/>
                            <xs:attribute name="level">
                              <xs:simpleType>
                                <xs:restriction base="xs:string">
                                  <xs:enumeration value="basic"
dcr:datcat="" ann:label="general setting"/>
                                  <xs:enumeration value="advanced"
dcr:datcat="" ann:label="setting specific to advanced users"/>
```

```
                                </xs:restriction>
                              </xs:simpleType>
                            </xs:attribute>
                          </xs:extension>
                        </xs:simpleContent>
                      </xs:complexType>
                    </xs:element>
                    <xs:element name="output"
dcr:datcat="http://www.isocat.org/datcat/DC-3804" minOccurs="0"
maxOccurs="unbounded" ann:documentation="description of some output file
used by recognizers">
                      <xs:complexType>
                        <xs:simpleContent>
                          <xs:extension base="xs:string">
                            <xs:attribute name="type"
dcr:datcat="http://www.isocat.org/datcat/DC-1978">
                              <xs:simpleType>
                                <xs:restriction base="xs:string">
                                  <xs:enumeration value="audio"
dcr:datcat="" ann:label="audio file, wav or other"/>
                                  <xs:enumeration value="video"
dcr:datcat="" ann:label="video file, mpeg or other"/>
                                  <xs:enumeration value="tier"
dcr:datcat="" ann:label="annotation tier file, xml"/>
                                  <xs:enumeration value="csvtier"
dcr:datcat="" ann:label="annotation tier file, csv"/>
                                  <xs:enumeration value="timeseries"
dcr:datcat="" ann:label="timed information file, e.g. numerical, xml"/>
                                  <xs:enumeration value="csvtimeseries"
dcr:datcat="" ann:label="timed information, e.g numerical, csv"/>
                                  <xs:enumeration value="auxiliary"
dcr:datcat="" ann:label="any other file"/>
                                  <xs:enumeration value="multitier"
dcr:datcat="" ann:label="multiple annotation tier file, xml"/>
                                </xs:restriction>
                              </xs:simpleType>
                            </xs:attribute>
                            <xs:attribute name="mimetypes"
dcr:datcat="http://www.isocat.org/datcat/DC-2571" type="xs:string"/>
                            <xs:attribute name="optional"
type="xs:boolean"/>
                            <xs:attribute name="info"
dcr:datcat="http://www.isocat.org/datcat/DC-2520" type="xs:string"/>
                            <xs:attribute name="level">
                              <xs:simpleType>
                                <xs:restriction base="xs:string">
                                  <xs:enumeration value="basic"
dcr:datcat="" ann:label="general setting"/>
                                  <xs:enumeration value="advanced"
dcr:datcat="" ann:label="setting specific to advanced users"/>
                                </xs:restriction>
                              </xs:simpleType>
                            </xs:attribute>
                          </xs:extension>
                        </xs:simpleContent>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                  <xs:attribute name="ref" type="xs:IDREFS"/>
                </xs:complexType>
```

```
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="CMDVersion" fixed="1.1" use="required"/>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

# Appendix III - Useful libraries

1. Java opencsv library (Apache 2.0 license) webpages:
    - Library: http://sourceforge.net/projects/opencsv/
    - Homepage: http://opencsv.sourceforge.net/
    - Javadocs: http://opencsv.sourceforge.net/apidocs/
2. Java javacsv webpages:
    - Library: http://sourceforge.net/projects/javacsv/
    - Javadocs: http://javacsv.sourceforge.net/
    - Examples: http://www.csvreader.com/java_csv_samples.php