94801 Villejuif Cedex

email: jacobson@idf.ext.jussieu.fr
boydm@vjf.cnrs.fr

List of unusual symbols: (font: ASAP SILSophia)

ŋ LATIN CAPITAL LETTER ENG
ʔ LATIN LETTER GLOTTAL STOP
ʊ LATIN SMALL LETTER UPSILON
ː MODIFIER LETTER TRIANGULAR COLON

Number of pages:
Number of tables: 0
Number of figures: 17

Keywords: speech annotation, linguistics, time-aligned text

FIGURE CAPTIONS

NOTE: The captions have been left in the body of the text to indicate the approximate placement of figures.

Linguistic documents synchronizing sound and text

Michel Jacobson, Boyd Michailovsky, John B. Lowe

ABSTRACT

The goal of the LACITO Linguistic Archive project is to conserve and disseminate recorded and transcribed oral literature and other linguistic materials in (mainly) unwritten languages, giving simultaneous access to sound recordings and text annotation. The project uses XML markup for the kinds of annotation traditionally used in field linguistics. Transcriptions are segmented into sentences (roughly) and words. Annotations are associated with different levels: metadata at the text level, free translation at the sentence level, interlinear glosses at the word level, etc. Time alignment is at the sentence (and optionally the word) level.

The project makes maximum use of standard, generic software tools. Marked-up data is processed using freely available XML software and displayed using standard browsers. The project has developed (1) an authoring tool, SoundIndex, to facilitate time-alignment, (2) a Java applet which enables browsers to access time-aligned speech, (3) XSL stylesheets which specify "views" on the data, and (4) a Common Gateway Interface which allows the user to choose documents and views and to enter queries. Current objectives are development of the annotation and software to facilitate linguistic research beyond simple browsing. Over 100 texts in 20 languages have been processed at the time of writing; some of these are available on the Internet for browsing and simple querying.

RÉSUMÉ

Le Programme Archivage du LACITO (Laboratoire de Langues et Civilisations à Tradition Orale du CNRS) a pour but la pérennisation, l'exploitation et la diffusion de documents linguistiques intégrant texte et son, en particulier les enregistrements faits et transcrits sur le terrain par les chercheurs du laboratoire. L'annotation (transcription, analyse, gloses interlinéaires, traductions) est balisée selon la norme XML et synchronisée, généralement phrase par phrase, avec l'enregistrement numérisé, pour donner accès simultanément au texte et au son.

Dans la mesure du possible des outils logiciels génériques et librement disponibles sont utilisés . Les documents produits sont consultés à l'aide des browsers les plus courants sur Internet. Le texte balisé est manipulé à l'aide d'outils génériques XML. Le programme a développé (1) un outil de création, SoundIndex, qui facilite la synchronisation du son avec le texte, (2) un applet Java qui permet aux browsers d'accéder au son, (3) des feuilles de style XSL qui définissent les "vues" sur les données, et (4) une interface (CGI) qui permet à l'utilisateur de choisir entre les documents et les vues disponibles ainsi que de formuler des requêtes, par exemple, pour chercher un mot particulier. Une centaine de documents dans une vingtaine de langues ont été préparés, dont certains sont disponibles sur Internet.

1. INTRODUCTION

The purpose of the LACITO Linguistic Archive project is the archiving of hundreds of hours of taped linguistic and ethnographic speech data, mainly in little-known and endangered languages, collected over the years by members of the LACITO (Langues et Civilisations à Tradition Orale) research group of the French CNRS (Centre National de la Recherche Scientifique). Apart from conservation, a major goal of the project is to make the recordings and their textual documentation available to researchers, including those who recorded them in the first place, using modern, computer-aided research methods. The present paper describes a system, using freely available software, for the creation of documents giving simultaneous access to sound and text, which can be browsed and queried over a network.

The project began in the early 1990s with the idea of digitizing original tapes and cassettes and conserving the recorded speech and other data on CD-ROM using the newly available CD-Recordable technology. From the outset, archiving the text annotation which accompanied the field recordings was a major concern. The annotation, and in particular the transcription of the recorded texts, is typically worked out in the field in consultation either with the original speakers or with other members of the same speech community. It is just as irreplaceable as the recordings themselves and requires far greater effort to prepare. Digitization makes it possible to archive both recorded speech and annotation together on the same media, eliminating the possibility of separation and partial loss. More positively, it facilitates the confrontation of the annotation with the recorded speech, and thus its correction and enrichment.

In the following pages we describe the ongoing development of a digital annotation for time aligned speech and of software for its exploitation. In section 2 we describe the traditional interlinear transcription of field recordings, the software which was available for

processing such documents at the time the project was conceived, and the reasons which led us to develop our own digital markup and software. In section 3 we describe our digital markup. In section 4 we describe the software tools which we have adapted or developed for authoring, browsing, and querying the archives. In section 5 we briefly describe two similar projects elsewhere. In section 6 we summarize the progress made and the lessons learned.

2. THE LEGACY DATA AND EXISTING SOFTWARE

2.1 Sound Recordings

The original sound recordings, whose possible deterioration was one of the motivations for the project, are on tapes and cassettes. These are digitized and conserved on CD-ROM, a routine technical operation which will not be discussed further here. The usual digital audio parameters of 44.1kHz sampling rate and 16 bit samples have been adpoted for digitizing and archiving the sound data, although not necessarily for its dissemination. The project software uses standard sound utilities and thus can handle a wide variety of digital sound formats.

2.2 Implicit structure of the legacy data

The structure of the traditional annotation, which will be familiar to linguists, is implicit in its typographical format (see Fig. 1). The basic annotation is the transcription of the original language. Typically, the transcription is divided into utterances, or intonation groups, or (roughly) sentences, and the sentences (as we will call them here) into space-delimited words. The sentences are numbered for reference; each one may or may not always begin on a new line. For our present purposes we will ignore other levels of structure. For example, blocks of transcribed text may be divided into paragraphs, marked typographically

by indentation, or into speaker turns. Typographical words may include prefixes, suffixes, clitics, etc., marked off by separators (e.g. hyphens in Fig. 1) and glossed separately.

In the literature on speech annotation, it is usual to consider transcriptions as source data, reserving the term annotation for analyses, translations, etc. The transcription of spontaneous speech in little-known languages, however, is built on a more or less shaky foundation of linguistic hypotheses, and we consider it as a kind of annotation. It is certainly not raw or unchanging data. It is true that much of the rest of the annotation can be regarded as annotation of the transcription.

Fig. 1: Transcription, interlinear gloss, and free translation of a Hayu narrative

Two types of translation are commonly supplied, aligned (at least logically) with the transcription at different levels: free translation at the sentence level and glosses at the word level or below. The free translations may be printed after each transcribed sentence, or, to facilitate connected reading, they may be grouped together at the bottom of pages or at the end of the text, in which case they are co-indexed with the transcription. Where word-glosses are supplied, these are placed on a line of their own and aligned under the corresponding words of the transcription. The interlinear glosses aid in the linguistic study of the text by indicating the semantic contribution of each word. The free, sentence-level translations, strung together and numbered in the original order, serve as an intelligible, if not always smooth or elegant, translation of the whole text. The literal word-glosses obviously lack this property: stringing them together in the order of the source language can be expected to yield word salad in the target language.

To recapitulate, the annotation of Fig. 1 is structured as a hierarchy of three levels, the text, the sentence and the word, with each category of annotation -- transcription, translation, and gloss -- associated with one level. Metadata concerning whole texts will be associated at

the text level (see below). The structure is hierarchical in that a text contains sentences and a sentence, words; it can be represented as a tree. At the same time, the annotation has linear structure in that the transcription units at each level are sequentially ordered. This linear structure will be put into correspondence with the temporal structure of the recorded sound below.

2.3 Existing markup and software

In this section we will review briefly the digitized annotations that were available at the time that the LACITO Archive project was conceived. Time-aligned digital formats have long been used in phonetics and speech engineering. Since these are often non-standard binary formats, and they are not designed for the kind of text annotation which we require, we did not consider them as relevant to our project. (For an inventory of all kinds of speech annotation, see the Linguistic Annotation web site.)

A few LACITO linguists had previous experience with digital formats and associated software packages which do not provide for alignment with recordings. We may briefly review these here, along with the reasons why we did not use them as the basis for the Archive project. Two systems, Lexware (Hsu 1989) and Shoebox (Buseman and Buseman 1998), use essentially identical "plain text" data formats which rely partly on explicitly labeled fields and partly on implicit structure (implicit hierarchy of fields, special separator characters, etc.) processed by the software. Thus in Shoebox, for example, the first sentence of Fig. 1 might be coded as in Fig. 2. The text sentence is marked up in a data record whose beginning is signalled by the label `\ref`. Each field of the record begins with a label explicitly indicating its nature -- line number, transcription, morphological analysis, gloss, free translation -- and ends with a line-break. Other aspects of the structure are implicit, for

example, that the nth space-delimited strings in the `\m` and `\g` fields annotate the nth space-delimited string in the `\t` field.

Fig. 2: An example of Shoebox annotation

Shoebox provides powerful tools for the entry and exploitation of multilinear text and for the simultaneous development of a lexicon; it is widely used by field linguists. It would no doubt be possible to incorporate the handling of time-aligned recordings into such a system, as has been done for CHILDES (MacWhinney 1995). Since these programs were conceived, however, the information industry has moved toward new standards in the areas of structured text markup, multi-byte character sets, access to multimedia, etc., all of which offer fundamental advantages in meeting the needs of the Archive project. We therefore decided to make a fresh start, based on the the widely-accepted norm for the definition of markup languages, SGML (Standard Generalized Markup Language, normalized as ISO 8879:1986). SGML would allow us to define a markup and document structure adapted to our needs, although the complexity and expense of processing software was something of a deterrent. Unicode (Unicode Consortium 2000) promised a solution to linguists' perennial font problems. Stylesheet definition languages existed for SGML and multimedia support was promised.

We were influenced by the *Guidelines for Electronic Text Encoding and Interchange* (TEIP3, Sperberg-McQueen and Burnard 1994) of the Text Encoding Initiative (TEI), which proposed SGML markup for a wide variety of document types used in literary and linguistic studies. We did not adopt the TEI piecemeal because the predefined markup went far beyond our needs in most areas while remaining rudimentary in the domain of speech annotation (TEIP3 Ch. 11).

As the project was getting started, the slightly simplified XML (eXtensible Markup Language) dialect of SGML was being developed for web applications in particular. When

the World Wide Web Consortium (W3C) adopted XML as a recommendation (Bray et al. 1998), it triggered widespread development of free and public domain software, a phenomenon that had previously occurred around the display language HTML (HyperText Markup Language, Raggett et al. 1999) but never around the original SGML. The Archive project (with a number of other academic projects, including a renewed TEI consortium) adopted XML markup in 1997.

3. THE LACITO MARKUP

3.1 Structured text and XML

In the section 2.2, the logical structure of the legacy annotation was derived from its typographical form. Now we will present XML markup which makes this structure fully explicit. For example, a gloss will be identified explicitly by delimiting tags containing a label of our choice -- `<GLS>` at the beginning and `</GLS>` at the end -- and not implicitly by its position on the page, font, style, etc. It will be grouped into a superordinate labeled structure with the corresponding word of the transcription. Text marked up in this way with labels identifying the functional, rather than the typographical, nature of each element is "logically structured text".

An XML document consists of structural markup and text data. It is organized into elements whose beginnings and ends are identified by user-defined structural tags. Between the start-tag and the end-tag, an element may contain other elements or text data or both, e.g., a gloss 'dog' might be marked up as `<GLS>`dog`</GLS>`. In addition to tags, the structural markup may include attribute-value pairs. For example, the language of the gloss 'dog' might be indicated by an attribute as `<GLS lang="English">`. Provision is made for reference to external resources, which may contain non-XML data such as sound or images. As mentioned, an element may contain other elements, but any contained element must be

entirely nested in the containing element; there can be no overlap. The structure of an XML document is thus equivalent to a rooted tree, the unique root element containing all the others.

The structural properties of an XML document or of a class of such documents can be declared in a Document Type Declaration (DTD). The DTD declares the elements that a document of the particular type may contain and the contents of each, defining the XML tree.

XML document-processing software begins by parsing the document to verify its "well-formedness", that is, its conformity to the syntactic rules of XML. A "validating parser" goes a step farther and verifies the conformity of a document to a particular DTD.

3.2 The LACITO DTD

The structure of the current LACITO XML markup is defined in a DTD which we will call the "LACITO DTD". The LACITO DTD may change to accommodate different versions of our markup. This has no effect on most of our software tools, including the authoring tool, as these are independent of the DTD and have only minimal expectations of XML documents apart from well-formedness. It is not an objective of the project to propose a unique LACITO DTD.

In the LACITO DTD's emulation of the legacy annotation, the text data is contained (not always directly) in elements labeled `TEXT`, `S`, and `W`, corresponding to the "text", "sentence" and "word" levels of structure. The `TEXT` element contains a `HEAD` (see below) and a `BODY`. The `BODY` is made up of `S` elements. Each `S` element is uniquely identified (via the "`id`" attribute) and contains the transcription (`TRANSCR`), made up of `W` elements, and a free translation (`TRADUC`). There may be more than one free translation, with the target language indicated by the value of an attribute. Each `W` element contains a `FORM` element -- the transcribed word -- and a `GLS` element -- the word-gloss. The words of the free translation are not explicitly marked up since there does not seem to be any reason to access them

individually. The HEAD contains metadata concerning the whole text, such as its title, the date, the speaker, the language, etc. Figure 3 shows the main elements of the LACITO DTD.

Fig. 3: Fragment of the LACITO DTD

Fig. 4 shows the markup of a fragment of a text in Hayu, a Tibeto-Burman language spoken in Nepal, conforming to the DTD of Fig 3:

Fig. 4: Markup of a linguistic text

3.3 Time-alignment markup

Sound recordings, by nature, can be segmented as finely or as coarsely as desired, certainly more finely than required to detect any linguistically significant element. The main decision to be made in aligning sound and text is the "granularity", that is, the length of the smallest text elements to be time-anchored, and hence the length of the smallest segments of the sound resource which can be accessed. Since the documents in question here are presented as connected text, it was decided to anchor units of connected text rather than, say, words or phonemes. On the other hand, because the languages of the texts are unfamiliar to most users, it was desirable to keep these units short. The simplest choice was to anchor the S-units, although anchoring to the word is also permitted by the DTD.

To preserve access to sequences of S's, or to the unbroken recording of the whole text, the original recordings are not segmented into separate S-sized sound resources. Instead, they are kept whole, with the segments corresponding to each S identified by their starting and ending time-offsets as measured from the beginning of the sound resource. The consequences of this are of practical significance because XML provides for only global reference to

external, non-XML resources. Accessing segments of a sound file has required some in-house markup and software development, as will be seen below.

The first requirement is identification of the sound resource. An XML document can reference external "entities", which may consist of either "parsable" (i.e. XML-formatted) or "non-parsable" (non-XML) data. The entity declaration, in this case of the non-parsable sound resource, specifies an entity name (used in future references), a physical location (computer file name and path), and a data-type (here non-parsable WAV data):

```
<!ENTITY SOEURS_SOUND SYSTEM "Hayu/SOEURS.wav" NDATA wav>.
```

The declaration may be located in the XML document which cites the entity in question, but we have chosen to centralize all such declarations in the DTD. This makes it possible to manage the archive sound resources globally, for example, switching between compressed and uncompressed versions in separate directories, without modifying the individual text documents, which use the unchanging entity names. It would have been possible to simply hard-code the filename and the path in the XML document:

```
<SOUNDFILE href="Hayu/SOEURS.wav"/>.
```

Such declarations, however, could not be globally managed and only implicitly indicate the nature of the data.

To mark up the alignment data, an `AUDIO` element is incorporated into each time-aligned element, with start- and end-time offsets as attributes. Fig. 5 shows how the `AUDIO` element would be added to the markup of the first `S` of Fig. 4.

Fig. 5: The AUDIO element

The oblique at the end of the empty AUDIO element obviates the need for an end-tag. We will discuss below how these offsets are interpreted.

As mentioned above, the sound recording has a completely linear structure which can be measured either by time or sample number, both external to XML. In the transcription, the start-offset of each S may typically coincide with the end-offset of the preceding S, but this is not necessary, and overlap is possible. Thus, in a conversation, speaker B might start before speaker A has finished. This can be coded as in Fig. 6.

Fig. 6: Speaker overlap reflected in time-offsets

In Fig. 6, the XML structure does not reflect the overlap, and indeed XML would not allow partial overlap between the end of <S who="A"> and the beginning of <S who="B">. Nevertheless, the temporal overlap in the recorded sound can be indicated as shown, because the values of these attributes are not XML structural elements and are not verified by the XML parser. If necessary, occurrences of overlap can be identified computationally from the attribute values.

Time-anchoring of elements at hierarchially different levels, for example S and W, does not imply partial overlap, so the XML markup can mirror the temporal structure as shown in Fig. 7. Note that the parser will not verify that W time-offsets are nested between those of the containing S.

Fig. 7: Multi-level time-alignment

Because the XML parser does not verify the values of the time offsets, it may be necessary to develop software to check for anomalies in time-anchoring for different sorts of documents. For example, one might wish to verify: (1) that the start offset of any unit has a lesser value than the end offset; (2) that start and end offsets of hierarchically inferior units are included within those of the containing hierarchically superior units; and (3) that the relation $end_n \leq start_{n+1}$ holds between the offsets of successive elements of the same type. This

last requirement would be relaxed for overlapping speech turns or for documents in which the order of elements is not necessarily fixed, for example, for a vocabulary list keyed to a recording elicited in a different order.

4. SOFTWARE TOOLS AND IMPLEMENTATION

4.1 Authoring tools

A major practical difficulty in preparing the archive documents is measuring the time-offsets required to align the transcriptions with the recordings and incorporating them into the markup. To facilitate this operation, which is not handled by standard software, an authoring tool, SoundIndex, was developed. SoundIndex incorporates a sound editor and a text editor. It plays a sound file (a wide variety of formats is supported, including WAV, AIFF, MPEG3, etc.) and simultaneously displays the waveform and a text, normally the transcription prepared by the linguist. It allows the user to mark on the waveform the start- and endpoints corresponding to each segment of the transcription and records the corresponding time-offsets.

The first version of this program, SoundIndex1, whose user interface screen is shown in Fig. 8, is not XML-aware. It accepts as input any text in which the segments to be time-aligned are delimited by a particular character (e.g. carriage return, full stop, or -- as in the figure -- opening square bracket) and creates a text table of start and end time-offsets (the window HAYU1.IDX in Fig. 8). The sound may be in any digital format supported by QuickTime, or it may be an audio CD track. After the offset table is created, Perl scripts are used to integrate the transcription, any other annotation, and the time offsets into an XML document conforming to the LACITO DTD. SoundIndex1, written in C++ for the Macintosh platform, is fully documented and freely available on the Archive project website. It has been

used by a half-dozen LACITO researchers. At least one other user, in Australia, was able to use the program "off the shelf" and reported the experience (Thieberger 1999).

Fig. 8: SoundIndex1

A new version, SoundIndex2, works directly with XML documents. The user specifies the label of the elements to be time-aligned. When any such element is selected and the corresponding part of the waveform identified, the program generates an `AUDIO` element with the appropriate time-offsets and incorporates it into the document (Fig. 9). SoundIndex2 uses a standard XML parser and can open any well-formed XML document; it does not attempt to validate conformance to a particular DTD. It reads the text from a standard Document Object Model (DOM site) interface and uses a Unicode-aware text editor to display the content, highlighting the tags. A preliminary version of SoundIndex2 is available on the Archive project website; it has been tested on Windows9X, NT, and Linux platforms.

The original prototype of SoundIndex2, developed in Java for platform independence and for compatibility with XML tools, has been delayed for lack of a suitable sound editor. The current version was developed in Tcl/Tk using the Snack sound extension (Sjölander 1999) and the Tk text widget, following the example of Transcriber (Barras et al. 1998 and this issue; see section 5 below).

Fig. 9: SoundIndex2

Although the process of time-alignment might seem tedious, field linguists who use SoundIndex generally profit from it to improve their transcriptions. The quick and precise access which the program affords to time-aligned texts greatly facilitates verification. An editing mode allows corrections to be entered directly into the text editor window.

3.2 Browsing tools

To facilitate dissemination of the project documents, the browsing system is built around CGI (Common Gateway Interface) scripts and standard web browsers. In principle, the documents are accessible to any client equipped with a standard browser, a soundcard, a program implementing the Java Media Framework API (JMF site), and a Unicode font. A typical display screen is shown in Fig. 10.

Fig. 10: Browsing the archive

The user chooses between a number of predefined "views" before opening a document for browsing. He may choose to see only the transcription, or the transcription and the free translation, or the transcription and the aligned word-glosses; he may choose the language of the translation if more than one is available. Once a document is opened, clicking on a displayed text element causes it to be highlighted and the corresponding sound to be played. The user may choose either to listen to one segment at a time or to the whole of the remainder of the text, by choosing between the "one sentence" and "continuous" buttons on the display (Fig. 10).

The system operates as follows. The client-server interface (CGI) requests the user to choose an XML document for browsing and a "view" on the data. The choice of a "view" is in fact the choice of an XSL stylesheet (eXtensible Stylesheet Language, Adler et al. 2000). A generic XSLT (XSL Transformations, Clark 1999) processor on the server uses the chosen stylesheet to format the required elements of the XML document as HTML, which is transmitted to the client machine. The HTML document contains references to the sound resource and to a playing applet, which are called by the client browser. It also contains JavaScript functions which handle user input (e.g. the mouse-clicks which select text for listening) and communicate with the applet. The applet handles access to segments of the

sound resource, a function which the browser cannot handle directly because it is not provided for in HTML.

Fig. 11: Part of an XSL stylesheet

In the remainder of this section, the operation of the browsing system is described in more detail. Fig 11 is an extract from one of our XSL stylesheets, whose function is to produce HTML. The first "rule" (contained in the first `xsl:template` element) instructs the processor, when the `BODY` element is encountered, to create an HTML ordered list (`"OL"`); it specifies that only `S` elements and their contents are to be considered. The second rule instructs it to create a new list item (`"LI"`) each time an `S` element is encountered. The list item contents are the transcribed word-forms (`TRANSCR/W/FORM`) of the current `S`; its attributes are (1) the `id` attribute of the current S and (2) an `onclick` attribute specifying that if the user clicks on the text of the `S`, a JavaScript function (`play(id)`) is to be called (see below). Application of the stylesheet (Fig. 11) to the sample document (Fig. 4) produces the HTML shown in Fig. 12.

Fig. 12: HTML resulting from application of the XSL style

The sound resource is accessed through a Java applet. When an HTML document is loaded, the applet is passed location of the sound resource, the identifiers (`id`) of all of the anchored segments of the text, and the start- and end-offsets corresponding to each. This is illustrated in Fig. 13. The applet is instructed to use the sound resource "`sample1.wav`". Since this is a sound resource (and not an image or video), it is not necessary to reserve screen space, so the applet screen window has the attributes `width=0 height=0`. Four pairs of start- and end-offsets are passed to the applet, corresponding to the HTML elements whose `id` attributes are "hayu1s1", etc. Given this information, the applet can either play the appropriate sound when passed an `id` attribute (text-driven mode) or identify the time-

anchored text segment corresponding to the current time-offset as the sound resource is played (time-driven mode).

Fig. 13: Sample HTML-formatted declaration and initialization of the Java applet

Text-driven synchronization, activated when the user chooses "one sentence" mode (Fig. 10), is the most straightforward. The text consists of a series of elements which may or may not be anchored. Each of the anchored elements has an `id` attribute permitting its unambiguous identification, and an `onclick` attribute which calls the JavaScript function `play(id)` when the user clicks on the text content of the element. The JavaScript function highlights the selected element and serves as an interface, passing a command to the Java applet which executes the sound resource. The applet looks up the corresponding start- and end-offsets and plays the sound, stopping when the end-offset is reached. The communication occurs from the browser to the applet, as indicated schematically in Fig. 14.

Fig. 14: Communication from script to applet in text-driven mode

Time-offset driven synchronization is used when the user has chosen "continuous play" mode (Fig. 10). When the user clicks on a location in the text, a JavaScript function `playFrom(id)` is activated and calls the applet, passing it the `id` argument of the containing time-aligned text segment. The applet finds the corresponding entry in its index list and builds a stack containing all of the events that will have to be generated as the sound resource is executed from the selected entry to the end of the resource. Since the sound, once started, takes care of itself, these events only control the text display. Independent "start" and "end" events are generated for the start and end time-offsets, respectively, of each time-anchored segment, and a "stop" event for the end of the sound resource. When the event stack

is complete and sorted by increasing time-offset, the applet orders the execution of the sound resource.

As the sound is played, the JavaScript function checks periodically for applet-generated events. Each time it is polled, the applet checks the current time-offset to see whether any new events need to be generated. If so, it calls the corresponding JavaScript functions, `startplay(id)` or `endplay(id)`, to advance the display. If execution of the resource is stopped, either because the end has been reached or because of a user interrupt, the applet empties its stack, generating "end" events for all current segments and a "stop" event. The script then stops checking for applet-generated events. The process is illustrated schematically in Fig. 15.

Fig. 15: Communication between script and applet in time-driven mode

Between the start and end events corresponding to a single segment, other events may occur, such as the beginning of another temporally overlapping or nested segment. The schema in Fig. 16 shows the order of events generated by the applet under these conditions.

Fig. 16: Order of execution of anchored elements

For playing single segments, the text-driven scenario described above has the advantage of simplicity. However, the time-driven method has the advantage of two-way communication between the applet and the browser, and we now use it for "single sentence" browsing as well. This is done in the following way. The JavaScript `playFrom` function passes the "single sentence" parameter to the applet. The applet constructs an event stack containing the events that occur between the beginning and the end of the aligned segment only. Applet-to-browser communication makes it possible, for example, to turn off highlighting when execution of the sound is terminated.

4.3 Managing and querying the archive

A major incentive for digitalization in the form of structured text is the possibility of data processing beyond browsing, in particular, management and exploitation of the archive as a text database. This normally would be done through a query language. The W3C has yet to publish even a draft recommendation in this area (Fankhauser 2000). In the meantime, we have used some of the currently implemented XML query language processors: XQL (Robie et al. 1998), XML-QL (Deutsch et al. 1998), LT XML (Thompson et al. 1997). In addition, many functions of a query language can be performed using XSLT to select and transform XML-coded information.

Our entire archive of about 100 time-aligned texts can be accessed through a single XML document of a type which we have called `ARCHIVE` in the LACITO DTD. An `ARCHIVE` contains elements of type `TEXT`. Our main archive document consists of (1) a declaration of each included `TEXT` as a parsable XML entity and (2) a reference to each declared entity. When this document is processed, the references are replaced by the entity contents, which is equivalent to having all of the texts in a single document. Bringing up the titles of all texts in a given language is accomplished by the following query, using the LT XML querying tool sggrep:

```
ARCHIVE/TEXT[lang='Hayu']/HEAD/TITLE
```

The result of such a query, formatted by XSLT and displayed, serves as the interface for choosing a text for browsing. A similar query could serve to dynamically constitute a corpus of texts for study and interrogation:

```
ARCHIVE/TEXT[lang='Hayu']
```

The document selection function can also be realized in XSLT (Fig. 17). Note that the operative part of the stylesheet, the value of the `select` attribute expressed in XPath notation (Clark and De Rose 1999), is almost identical to the sggrep query.

Fig. 17: An XSL stylesheet to extract the titles of a class of documents

We have used XQL to implement a prototype word-searching application. A straightforward CGI permits the user to enter a "word" or a regular expression. This is passed to the query language, the response is formatted, and a KWIC (KeyWord In Context) listing of all `S` units in which a matching string occurs as a `W/FORM` is displayed. Because the response includes the sound resource declaration, and the `S` units are returned with their `AUDIO` elements, the user can click on any `S` in the KWIC listing and hear the sound. The function which we wrote to permit regular expression matching on Unicode text in XQL became obsolete when a similar function was included in the program distribution.

To facilitate user entry of Unicode strings, we have designed an alternative interface which constructs and displays an alphabetized index of all the words in a text, so that the user can simply click on a word for searching. This interface is useful for simple word-searches on the relatively small corpora field linguists commonly work with.

Finally, a concordancing tool has been prototyped which makes an alphabetized KWIC index of all the words in a text, showing left and right context for each occurrence of each word. The sort order can be defined to suit a particular language and transcription, with either right or left context as a secondary field. Clicking on a line of the concordance brings up the sound recording of the containing `S`. The concordancing tool prototype was written in Perl to take advantage of a sort-defining tool we had previously written. Since non-standard sort orders are very useful for linguists, we are working on a similar function to incorporate into XSLT.

In addition to new tools, improvement in the annotation is required to facilitate linguistic analysis. For example, because users typically wish to search for either roots or affixes, not affixed or inflected word forms, we have begun including a morphological transcription, subordinate to the word level and in parallel with the word transcription, in some texts (cf. the line labeled "\m" in Fig. 2). This annotation will also be useful for accessing online lexicons from text documents. XML-coded lexicons of Hayu and Limbu (Tibeto-Burman languages of Nepal) are currently under development.

5. RELATED PROJECTS

A number of other projects have taken a direction similar to ours in recent years. We will briefly describe two of these here. A complete list of projects, old and new, in the speech annotation area can be found on the Linguistic Annotation site.

Transcriber (Barras et al. 1998 and this issue) is an authoring tool similar to SoundIndex, in particular to SoundIndex2. The differences between the programs are closely tied to their initial environments and purposes. Transcriber was originally written for a production environment in which operators would rapidly transcribe and time-align radio broadcasts in standard spelling, with some extra annotation of speech turns, background noises, etc. The annotation is recorded as XML conforming to a fixed DTD. Thus the LACITO markup could not be accommodated. However, if a LACITO linguist wished to enter a text transcription and align it using Transcriber, the result could be converted, using an XSLT stylesheet, to markup conforming to the LACITO DTD.

MATE (Multilevel Annotation, Tools Engineering, Dybkjær et al. 1998, McKelvie, this issue) proposes a common XML exchange format for a number of existing corpora with different codings and purposes, as well as a software workbench of authoring, querying, and

display tools. It is not tied to a particular DTD. Linguistic analysis is one of the main purposes for which it is designed.

Although the MATE internal data representation is based on the "annotation graph" model of Bird and Liberman (1999, this issue), most MATE processing comes down to the processing of XML documents. MATE has adopted "standoff markup" in its XML data representation, as outlined in the TEI guidelines and the Corpus Encoding Standard (CES, Ide and Priest-Dorman 1999). This makes it possible to encode multiple intersecting hierarchies of annotation (e.g. syntactic and prosodic annotations, or syntactic annotations reflecting different analyses) in separate XML documents which are multiply linked to a single transcription document. (The relatively simple LACITO morphosyntactic annotation has not required such a mechanism as yet.) As long as only one hierarchy is used at a time, standoff markup should be able to be processed as standard XML.

The MATE workbench, according to the published description (McKelvie, this volume), is particularly interesting in that it is designed to take into account the DTD of a speech annotation document and to create, or facilitate the creation of, an editor/authoring tool adapted to the DTD. Thus it steers a middle course between SoundIndex, which is flexible but ignores the DTD, and Transcriber, which requires a fixed DTD.

Neither Transcriber nor MATE attempts to make time-aligned documents accessible to standard browsers; both integrate the authoring and browsing functions. Both projects are considering adopting generic XML processing tools: at present Transcriber uses its own parser, and MATE its own stylesheet processor (MSL) and its own query language (Q4M). The use of standard XML makes interoperability with the LACITO or other markup a possibility.

6. CONCLUSIONS

The LACITO Archive Project has developed a method of digitizing the traditional linguists' annotation of oral texts in XML, including time-alignment with digitized recordings, and a system for browsing and querying the resulting documents over the Internet. More than a hundred texts in a score of languages have been prepared at the time of writing; some of these have been made publicly available on the project web site. About 10 hours of texts in 14 languages of New Caledonia have been prepared for the multimedia library of the Tjibaou Cultural Center in Nouméa, New Caledonia. Texts will continue to be prepared, and most will be made publicly available as intellectual property issues are resolved.

A "LACITO DTD" has been developed to describe the annotation, which varies somewhat from corpus to corpus. The annotation essentially identifies utterances and words, and in some cases morphemes, and makes them available for processing and querying. It associates translations with sentences, and glosses with words and morphemes. Time-anchoring of sentences and words is provided for.

As explained above, it is not the aim of the project to propose a single, standard annotation. The text corpora are in typologically diverse languages and have been prepared by linguists with differing research interests. Under these circumstances it is illusory to imagine that agreement could be reached on a single, detailed markup. Although it may be possible to maintain a single DTD to cover the annotations, this, too, could become undesirable if it makes it more difficult to ensure the coherence of any one corpus annotation or to adapt tools to process it. In our view, standardization is to be sought in adherence to the principle of logically structured text, which makes it relatively easy to transform one markup into another, rather than in particular markup conventions.

The project has developed a number of software tools. An authoring program, SoundIndex, has proved its value as a tool for sound-aligning of legacy data, but it has only rudimentary XML editing capability. Such capacity is needed especially for entering new annotation, but also for correcting existing annotation, including transcriptions. (Linguists are invariably led to improve their transcription when they have time-aligned access to the recordings -- proof, if one were needed, of the scientific value of time-aligned documents.) The principle of generating editors adapted to particular DTDs, suggested by the MATE project, seems particularly worth pursuing for this purpose.

The Archive Project browsing and interrogation systems are entirely separate from the authoring tool. Linguistic documents prepared by the project are browsed and queried using standard browsers through a standard web interface, facilitating their dissemination. The browsing system is relatively stable, but the data management and querying software is in an early stage of development, the latter in part because of a lack of defined standards and hence of standard tools. General searching, word-indexing, and concordancing modules have been developed in prototype form.

From the technical point of view, the key to the Archive Project has been the adherence to standards, XML in particular, and the use wherever possible of generic tools which are maintained by others to keep up with changing hardware, web standards, etc. Reliance on generic software allows us to concentrate our limited technical resources on the development of specifically linguistic tools. We are able to choose between several different generic XML parsers and XSL processors, changing them as improved versions are distributed. The fact that many generic tools are freely available is an obvious advantage; the fact that they are open-source has been helpful in that it has allowed us to add specific modules to them while retaining their generic functionality. We have taken advantage of this

to incorporate regular expression matching into a query processor and are planning to incorporate complex sort order parametrization into XSLT.

The LACITO Archive Project is one of a number of projects which are converging on common principles: the use of logically structured markup, in particular XML, at least as an exchange and processing format, and the use, to the extent possible, of generic software so that processing is accomplished through high-level stylesheets and queries. The LACITO project has benefited from open-source software, and all programs developed by the project are open-source. As more projects adhere to common standards, we expect that distributed development and interoperability of software for field linguistics and other specialities will become a reality.

ACKNOWLEDGEMENTS

1. The initial prototype documents of the LACITO Archive Project were time-aligned using a Hypercard program and marked up in HTML by J.B. Lowe in 1996. Michel Jacobson wrote the current versions of SoundIndex, and all other programs, XSL stylesheets, etc., currently used by the project. The project received support under the Language Engineering program of the department of Human and Social Sciences (SHS) of the CNRS. Texts in New Caledonian languages were prepared by Oceanist linguists of the LACITO under a contract between the Agence du Développement Culturel Kanak, Nouméa, and the LACITO.

REFERENCES

Adler, Sharon, Berglund, Anders, Caruso, Jeff, Deach, Stephen, Grosso, Paul, Gutentag, Eduardo, Milowski, Alex, Parnell, Scott, Richman, Jeremy, Zilles, Steve, eds., 2000. Last Call Working Draft for the W3C Extensible Stylesheet Language (XSL) Version 1.0. W3C working draft. March 28, 2000. (http://www.w3.org/TR/WD-xsl)

Barras, Claude, Geoffrois, Edouard, Wu Zhibiao, Liberman, Mark, 1998. Transcriber: a Free Tool for Segmenting, Labeling and Transcribing Speech. Proceedings of the First International Conference on Language Resources and Evaluation (LREC), Granada, Spain, pp. 1373-1376. (http://www.etca.fr/CTA/gip/Projets/Transcriber/Index.html)

Bird, Steven, Liberman, Mark, 1999. A formal framework for linguistic annotation. Technical Report MS-CIS-99-01, Computer and Information Science, University of Pennsylvania. (http://xxx.lanl.gov/abs/cs.CL/9903003)

Bray, Tim, Paoli, Jean, Sperberg-McQueen, C. M., eds., 1998. Extensible Markup Language (XML) 1.0. W3C Recommendation. (http://www.w3.org/TR/REC-xml)

Buseman, Alan, Buseman, Karen, 1998. The Linguists SHOEBOX. Summer Institute of Linguistics. (http://www.sil.org/computing/catalog/shoebox.html)

Clark, James, ed., 1999. XSL Transformations (XSLT) version 1.0. W3C Recommendation. (http://www.w3.org/TR/1999/REC-xslt-19991116)

Clark, James, DeRose, Steve, 1999. XML Path Language (XPath) Version 1.0. W3C Recommendation. 16 November 1999. http://www.w3.org/TR/xpath.

Deutsch, Alin, Fernandez, Mary, Florescu, Daniela, Levy, Alon, Suciu, Dan, eds., 1998. XML-QL: A Query Language for XML. Submision to the W3C XSL Working Group. (http://www.w3.org/TR/1998/NOTE-xml-ql-19980819.html)

DOM. http://www.w3.org/DOM/Overview.html

Dybkjær, Laila , Bernsen, Niels Ole, Dybkjær, Hans, McKelvie, David, Mengel, Andreas., 1998. The MATE Markup Framework. MATE Deliverable D1.2. (http://mate.nis.sdu.dk/information/d12/)

Fankhauser, Peter, Marchiori, Massimo, Robie, Jonathan, eds., 2000. XML Query Requirements. W3C Working Draft. 31 January 2000. http://www.w3.org/TR/xmlquery-req

Hsu, Robert, 1989. Lexware Manual. Second Edition. Linguistics Department. University of Hawaii. ms.

Ide, Nancy, Priest-Dorman, Greg, 1999. Corpus Encoding Standard. Document CES 1. Version 1.5. (http://www.cs.vassar.edu/CES/)

Java Media Framework. htp://java;sun.com/products/java-media/jmf/

LACITO Archive Project. http://lacito.vjf.cnrs.fr/ARCHIVAG/ENGLISH.htm

Linguistic Annotation. http://morph.ldc.upenn.edu/annotation/

MacWhinney, B., 1995. The CHILDES system. In: Ritchie W., Bhatia, T., eds., Handbook of Language Acquisition. Academic Press, New York. (http://www.childes.psy.cmu.edu).

Raggett, Dave, Le Hors, Arnaud, Jacobs, Ian, eds., 1999. HTML 4.01 Specification. W3C Recommendation. 24 December 1999. http://www.w3.org/MarkUp.

Robie, Jonathan, Lapp, Joe, Schach, David, eds., 1998. XML Query Language (XQL). Proposition to the W3C XSL Working Group. (http://www.w3.org/Style/XSL/Group/1998/09/XQL-proposal.html)

Sjölander, Kåre, 1997-99. The Snack Sound Extension for Tcl/Tk. (http://www.speech.kth.se/snack/)

Sperberg-McQueen, C. M., Burnard, Lou, eds., 1994. Guidelines for Electronic Text Encoding and Interchange. (TEI P3). Electronic Book Technologies, xxx. [Revision date 16 May 1994. On CD-ROM. Citations are by section number.] (http://www-tei.uic.edu/orgs/tei/)

Thieberger, Nick, 1999. Using SoundIndex to transcribe fieldnotes. (http://www.linguistics.unimelb.edu.au/CALW/sindex.html)

Thompson, Henry, Tobin, Richard, McKelvie, David, Brew, Chris, 1997. LT XML Software. (http://www.ltg.ed.ac.uk/software)

Unicode Consortium, 2000. The Unicode Standard, Version 3.0. Reading, MA. Addison-Wesley.