Archiving Language Resource Objects in XML: Experiences with TAMINO

Dafydd Gibbon, Thorsten Trippel, Ben Hell

Universität Bielefeld, Europe

{gibbon|ttrippel|ben}@spectrum.uni-bielefeld.de

Overview

- Archiving in XML
- Language Resources
- Getting abstract: types of Resource Object
- Abstract Resource Object implementation in XML
- Getting Practical: the ModeLex application
- Using an XML database: TAMINO

  Procedure - Database creation - Corpus data stored in the file system - Using a DBMS for storing Resource Objects - Selected use cases - Querying - Signal processing
- Conclusion: evaluation and further work

Archiving in XML

Text Technological Modelling of Information

Background: projects

- Ega (2001)
- ModeLex (2001-...)
- ABUILD (2002-...)
- LLSTI (2003-...)

Goal: specifying a DBMS for Resource Object storage

Resource Objects:

- *General Resource Object* (GRO, linguistic data type)
- *Specific Resource Object* (SRO, instance of GRO)
- *Abstract Resource Object* (ARO, abstract data structure)
- *Implementational Resource Object* (IRO, PL/KRL data structure)

Language Resources

Written texts, dialogue transcriptions

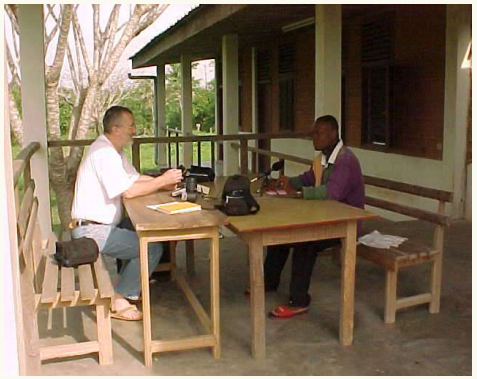
Annotations

- time-stamped transcription
- marked up written text & transcription

Signal recordings

- audio, video, laryngograph (electroglottograph), airflow, ...

Lexical information



Multimodal resource search:

- structuring with XML
- storing
- accessing
- updating

Getting abstract: types of Resource Object

General Resource Object (GRO, linguistic data type)

Specific Resource Object (SRO, instance of GRO)

Abstract Resource Object (ARO, abstract data structure)

strings - string sequences - structures over strings - lists - tables - DAGs - CGs - numbers - ...

Implementational Resource Object (IRO, PL/KL data structure)

TREES: typically for constituent structures and taxonomies

TABLES: typically for lexica and paradigm tables

DAGs: typically for (almost) anything ☺

Abstract Resource Objects and XML IROs

XML abstract syntax defined as recursive ternary relation...

OBJECT = string

OBJECT = {x: x = <typename, AVS, OBJECT^{+}>}

... can only define tree structures: $a^n b^n$ (Type 2, CF L)

Not defined in XML syntax:

For embedded tables further constraints necessary...

... general indexing needed: $a^n b^n c^n$ (Type 1, CS L subset)

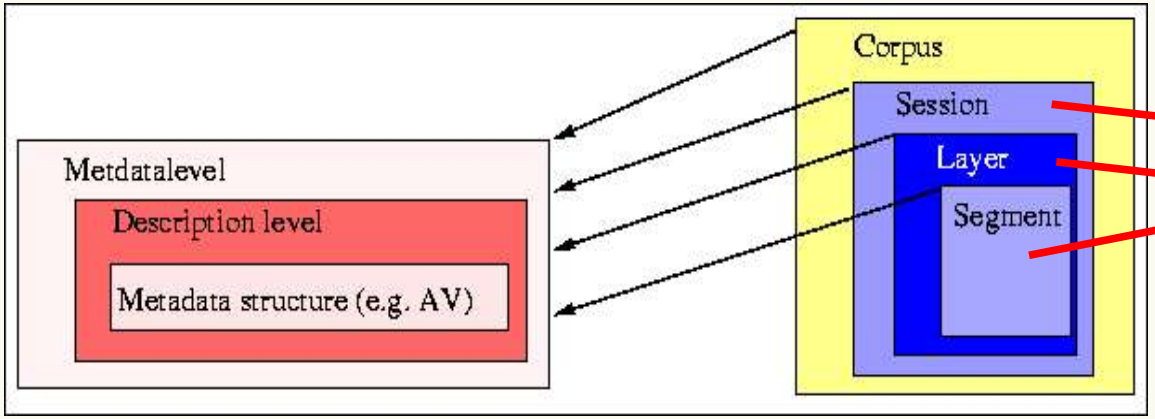
For general graphs, networks, *semantic* extension needed:

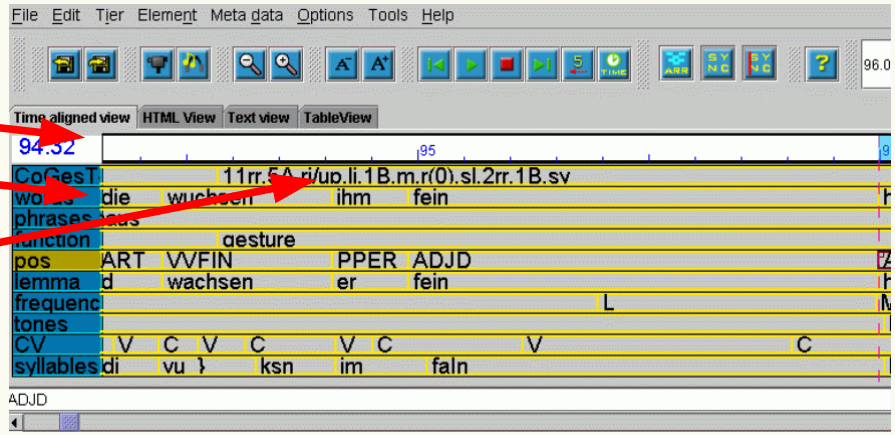
pointer structures permit extension beyond tree structures.

Thus: access tools must be more powerful than XML syntax requires.

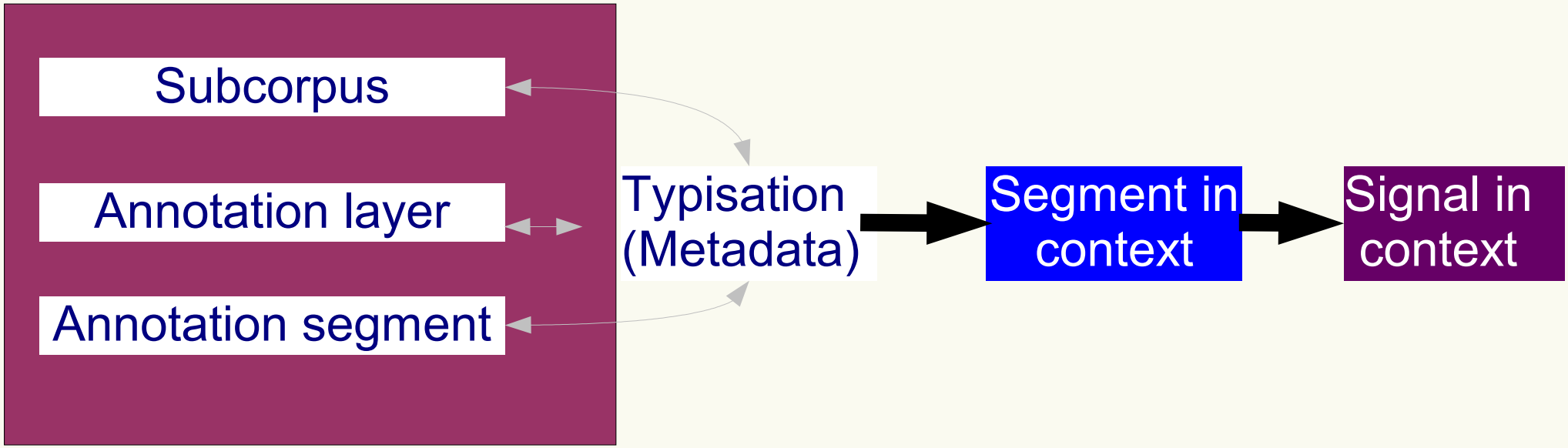
Getting practical: the ModeLex application

Corpus layers and metadata layers:





Search application:



Preliminaries: XML format normalisation

Text Technological Modelling of Information

Corpus format: depends on application
(WAV; praat, esps-waves+, TASX, ...)

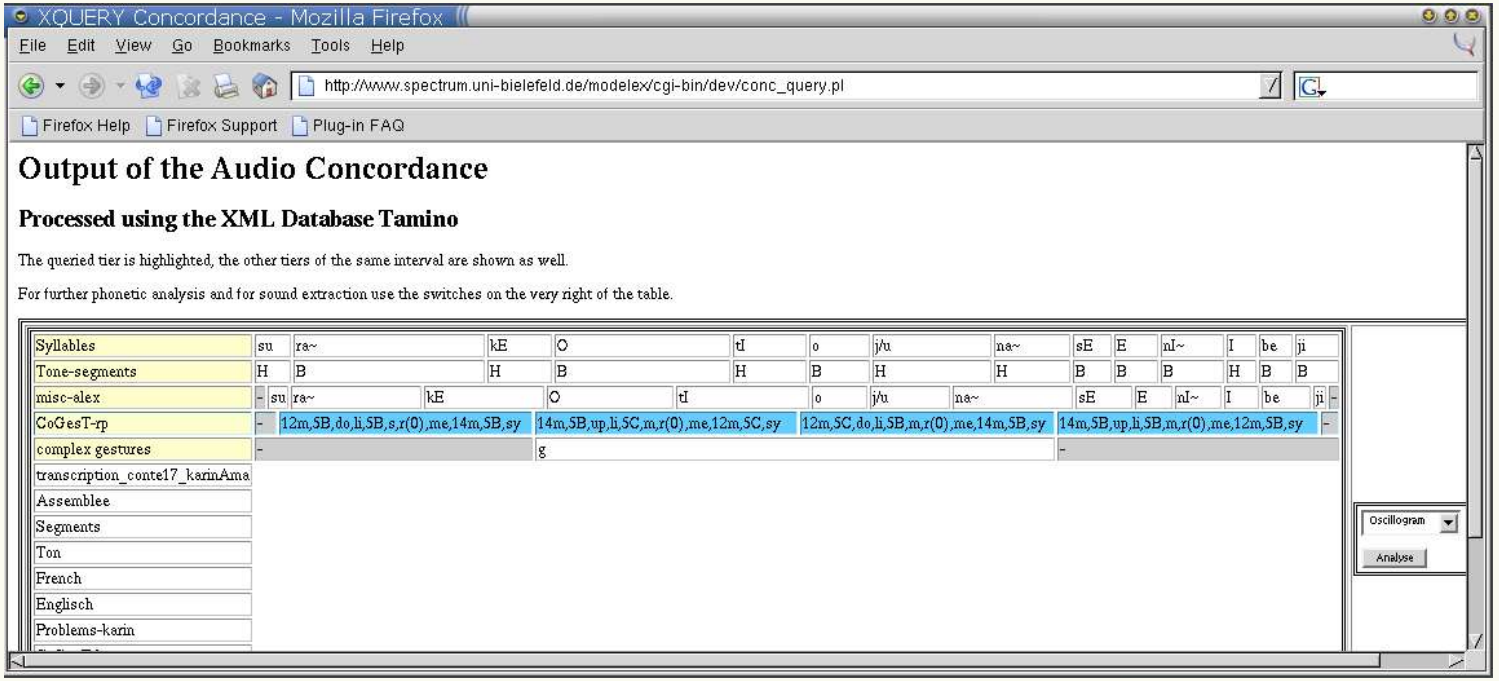
Normalization:
XML format
Preservation of all bits of information from source
metadata
timestamps
technical information
Time Aligned Signal eXchange format (TASX)

Grammar normalization: DTD to XSchema conversion

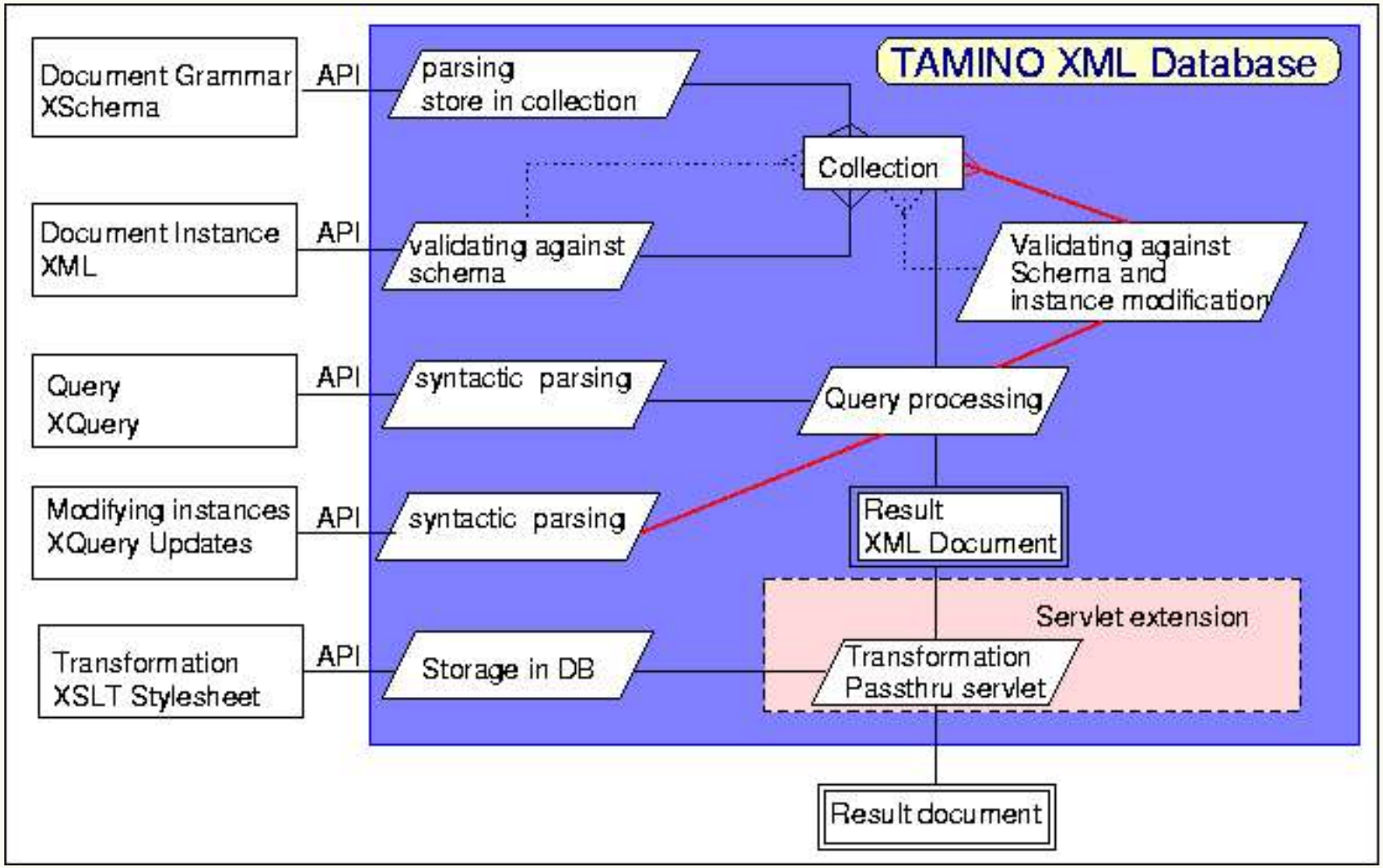
Use case: Multimodal concordance

Functional requirements specification:

Input: <searchkey, <recording, annotation>>
Output: subset of <recording, anotation>
- matching search key + context-tier
- corresponding to output format filters
(tiers, length, signal transformation)



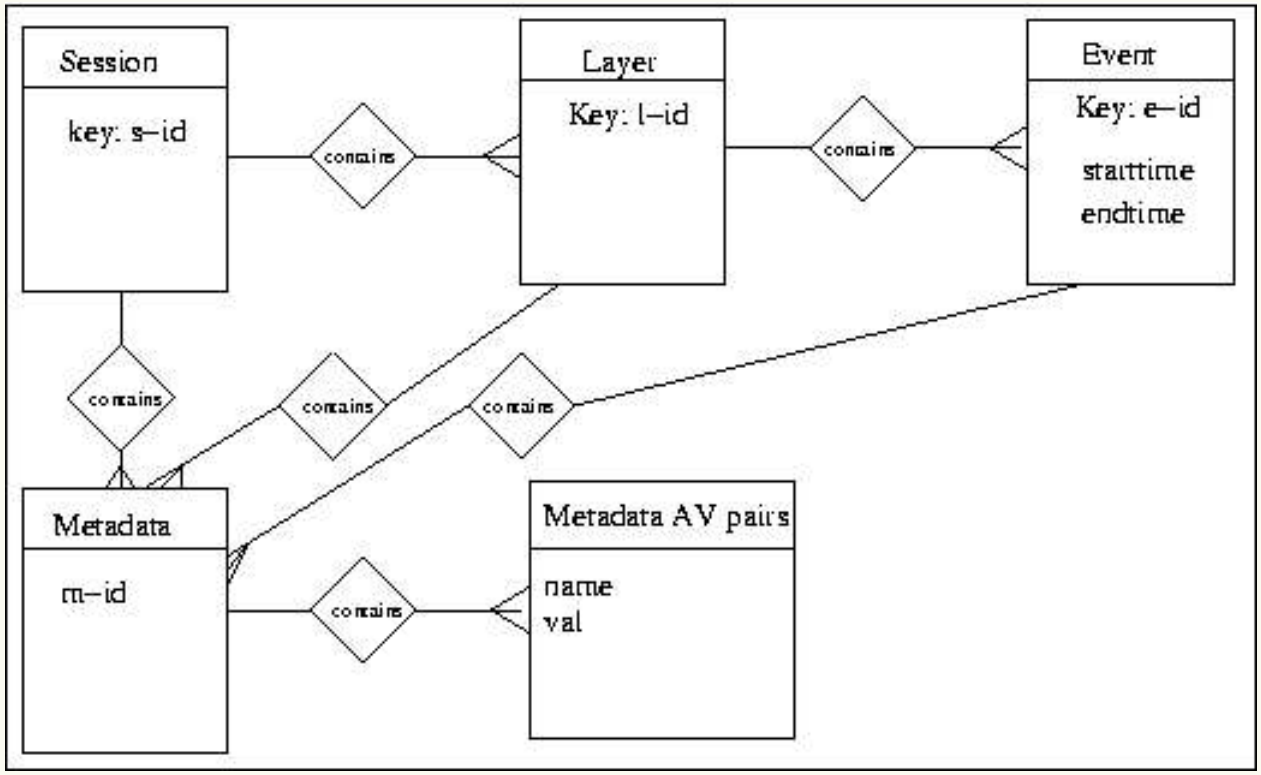
Design: signal concordance



Implementation: TAMINO XML DBMS - 1

Options:

1. data on file system:
   - command line access
   - easy to manipulate
   - selection complex
   - performance with large repositories



2. storage in TAMINO
   - create DB (Tamino Software Management Hub)
   - create "collection" (Tamino: Schema Editor)
   - insert schema (Tamino: Schema Editor)
   - insert document instances (Tamino: Schema Editor, Tamino Interactive Interface,...)

Implementation: TAMINO XML DBMS - 2

Traditional tools:
- file system + ad hoc tools

XML command technologies:
- filesystem based XQuery
  - saxon XQuery tool /Java Library
  - exquisit: GUI for saxon

Tamino based tools:
- Tamino interactive interface, webinterface
- Tamino XQuery (Windows Application)
- Tamino Java API
- Perl API: any Perl program, e.g. browser based GUIs

Implementation: TAMINO XML DBMS - 3

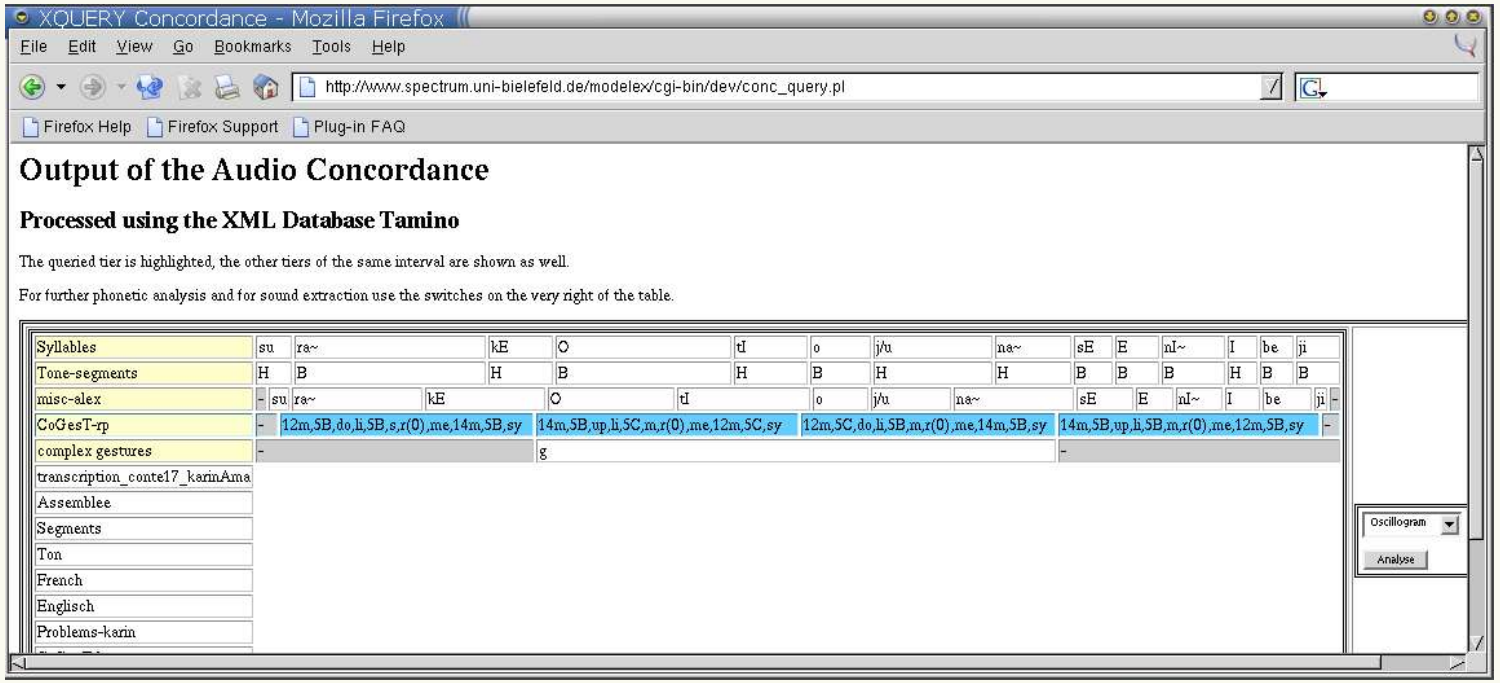
Text Technological Modelling of Information

Access:

based on XQuery
unit selection using metadata AND annotation segment key
context selection on same tier OR parallel tiers

based on

time interval → XQuery arithmetic in Tamino
sibling access → available in saxon, not in Tamino



Implementation: TAMINO XML DBMS - 4

Audio:

- Selected interval based on time stamps
- Further analysis possible if lossless compression files:
  - spectrogram, oscillogram, formant analysis, ...
- Fast (almost real time): praat scripting + sox

Gibbon and Trippel 2001: *Portable Audio Concordance System.* TR-UBI

Video:

- Audio in principle as above
- Granularity:
  - frame based, not sample-based
    - technical restrictions: keyframe rate
- Time consuming: no real time processing

Conclusion: evaluation and further work

Summary:

- Proof of concept for TASX audio corpus
  - Tamino, Perl
  - Audio signal processing: PAX modules, based on Praat

XQuery selection: corpus - subcorpus - layer - segment

To do:

- GUI not fail-safe (fails if metadata incomplete)
- Inconsistency potential in file storage of signal recordings
- Optimisation of XQuery vs. XSLT for formatting

http://www.spectrum.uni-bielefeld.de/modelex/implementation/concordance.html