6th European Conference on
Speech Communication and Technology
(EUROSPEECH'99)
Budapest, Hungary, September 5-9, 1999

# RELATIONAL vs. OBJECT-ORIENTED MODELS FOR REPRESENTING SPEECH: A COMPARISON USING ANDOSL DATA

*Toomas Altosaar[1], Bruce Millar[2], Martti Vainio[3]*

[1]Helsinki University of Technology, Acoustics Laboratory
[2]Australian National University, Computer Sciences Laboratory
[3]University of Helsinki, Department of Phonetics

## ABSTRACT

Much effort has been expended by the speech community in designing and producing speech corpora and databases. However, considerably less thought has been given to the type of database model that best represents speech for optimal database access. This paper reviews an established database model, the relational database management system (RDBMS), as well as an emerging one, the object-oriented database management system (OO-DBMS). Both model types are compared by addressing identical queries to the same data that is separately represented under these respective systems. This is accomplished using an identical source of richly described speech data from ANDOSL (Australian National Database of Spoken Language). By identifying the strengths and weaknesses of both approaches new models can be defined that incorporate the best of both existing systems enabling more fluent and extensive extraction of knowledge from speech databases.

10.21437/Eurospeech.1999-223

## 1. INTRODUCTION

Large data corpora of spoken language represent the paramount resource for the rapidly evolving field of speech technology. The extraction of information and knowledge from corpora has become more important recently as speech is modelled in more exacting detail for differing applications, e.g., word recognition, prosody generation, phonetic analysis, etc. Without fluent and precise access to such systems the development of robust analysis, synthesis, and recognition strategies would not be possible. Much effort has been expended on designing and producing speech corpora and databases. However, considerably less thought has been given to the type of database model that best presents a natural interface to speech for optimal database access. Given powerful and flexible means to index and retrieve speech data, more of the hidden potential existing in these collections of spoken language can be revealed and put to use.

Most speech processing systems rely on some form of a structured database access mechanism. Since the relational database management system (RDBMS) is a well-established paradigm and widely used in many non-speech fields it has been adopted to model spoken language as well. Another database paradigm, the object-oriented database management system (OO-DBMS), more recently developed and less known than RDBMS, is also able to represent speech data. This paper presents both the RDBMS and OO-DBMS models and compares the effectiveness of their representations using an identical corpus of speech data. By investigating issues such as query complexity, flexibility, system resource requirements, scalability, and database access efficiency, the benefits and drawbacks of both paradigms for representing speech data are exposed.

Speech data from the Australian National Database of Spoken Language (ANDOSL) [1] is used as source data for both the RDBMS and OO-DBMS representations. Besides including speech waveforms and phonemic annotations, ANDOSL includes a detailed description of recording environments, speaker characteristics, speaker histories, data format, data anomalies, date, time and place of recording, speaking tasks, and signal quality. For advanced studies this type of information can be crucial and needs to be carefully modelled in a database environment as well.

Relational databases comprise multiple tables in each of which the values of a number of data keys are defined for each data entry. The NSLD scheme, developed at the Australian National University [2], models ANDOSL data in a RDBMS framework where queries can be performed using a string matching formalism comparing the values of selected data keys within and across tables.

In an object-oriented environment data are modelled as objects. Object-oriented databases store objects as well as their relationships with other objects. For example, objects such as speech signals, speakers, recording environments, equipment, annotations, etc., are selectively linked to one another creating a structure where database access is accomplished by structure traversing search functions revealing desired contexts. QuickSig [3], an object-oriented signal processing environment, developed at the Helsinki University of Technology, is used as the OO-DBMS framework in this comparison.

This paper reviews the types of data existing in ANDOSL and their intrinsic relationships. Both RDBMS and OO-DBMS implementations are formed and subjected to identical database access queries.

## 2. RDBMS vs. OO-DBMS

Relational databases are centred on the idea of two-dimensional tables which are called relations. These tables are isomorphic to mathematical relations which have a solid theoretical foundation. Therefore, relational databases are well understood and are also technologically mature and stable.

However, there are certain data structures that either don't fit well into relations; or that, when shoehorned into relations, do not lend themselves well to querying [4]. Analysis of spoken language often entails complex linguistic structures or objects that cannot be easily handled in a RDBMS, e.g., signals, arrays or lists of

other objects. The description of spoken language represents a potential area where object orientation can be applied to reap the benefits of *abstraction*, *encapsulation*, and *modularity* [5]. Object orientation is one solution for the above-mentioned mismatch between the application and the database. In an OO-DBMS system the application and the database can be brought closer to each other and the programming language of the application can be integrated with the database's data manipulation language. Object-Oriented databases lack the mathematical foundation that relational databases possess and are not as well understood theoretically. Nevertheless, they support user-defined arbitrarily complex, data types. They can also deal with nested objects and they preserve strong typing across the interface between the application and the database. This eliminates the so-called impedance mismatch caused by the narrow and limited interface between a RDBMS and the application [4].

## 3. ANDOSL DATA CORPUS

The ANDOSL data corpus has several components including isolated words, phonemically-rich sentences, and spontaneous speech. These data were collected from a total of 264 speakers in either a single-speaker reading situation or dual speaker dialogue situation [1,6,7]. Each speaker contributed on the order of 80 isolated words including spoken digits, common consonantal context vowel exemplars, and some other key words. Native speakers of Australian English contributed 200 sentences designed to include all phonemes and all second-order pairings of phonemes of the language. Migrant speakers, who comprise some 22% of the population, contributed 50 sentences that were derived from the 200 but whose second-order pairings were restricted to at least one exemplar of each pair of broad phonemic classes. Furthermore, each speaker was paired with another speaker for two recordings of structured but free flowing dialogue using the MAP task. For the queries in the following sections, data was restricted to the YB_Sentences CDROM which includes 200 sentences each from six male and six female speakers for a total of 2400 utterances.

## 4. ANDOSL IN AN RDBMS FRAMEWORK

Access to ANDOSL via database systems using relational structures has been explored in two ways:

i) At Macquarie University the linguistic structure of the data has been modelled using the EMU system [8] which comprises a layered relational structure ranging from an acoustic-phonetic level, up via a phonemic level to a syllabic level, and eventually reaching an orthographic level. This structure may be traversed in either direction.

ii) At the Australian National University (ANU) the description of the source and quality of the data files comprising the speech signals has been modelled using the regular tables of an RDBMS system. It is this form whose raw data is distributed on the ANDOSL CDROMs.

The RDBMS system at ANU is based around a Notional Spoken Language Description (NSLD) scheme first described in [2]. NSLD has developed towards a comprehensive scheme [9] and thus the ANDOSL RDBMS is a subset. It has been used extensively for the selection of data for issuing on CDROMs and has been made available through a limited set of WWW forms (http://andosl.anu.edu.au/andosl) to users of the data who are searching for something specific within it, or who wish to request further ANDOSL data that was not included in the CDROM releases. Some additional tables have been added to service these particular needs such as a QUALITY and an ANNOTATION table which provide information about the signal quality of each signal file and the presence of phonemic annotation data. Table 1 lists the RDBMS tables.

| | |
|---|---|
| DDF(SOURCE) | Links Data, Speaker, Material ids; date, time, length |
| FORMAT | Extensive information on file format |
| LOCATION | Extensive information on file location |
| TRANSFORM | Information on post-digitisation transforms |
| DERIVED | Information of additional related data |
| ANNOTATION | Info. about the method of phonemic annotation |
| ENVIRONMENT | Information on the recording environment |
| ANOMALY_CODES | Info. on anomalies occurring in assembling the data |
| QUALITY | Info. on signal level, distortion, background noise |
| SPEAKER | Extensive static information about the speaker |
| SPEAKER_HISTORY | Episodic info. about the speakers life history |

**Table 1.** RDBMS tables currently implemented at ANU for the description of ANDOSL.

| | | |
|---|---|---|
| SID: | <speaker_identifier> | |
| %---------- MEASUREMENT METHODS | | |
| MET: | <filename.met> | % methods for all anatomic measures |
| %---------- PERSONAL/PHYSICAL | | |
| SEX: | <sex_of_speaker> | % "m" or "f" only |
| YOB: | <year_of_birth> | % 'four digit number' |
| LEV: | <education_level> | % "primary", "tertiary", etc |
| HGT: | <height> | % 'four digit number' - unit = mm |
| WGT: | <weight> | % 'three digit number' - unit= kg |
| LVT: | <length_of_vocal_tract> | % 'three digit number' - unit= mm |
| COH: | <circumference_of_head> | % 'three digit number' - unit= mm |
| ECD: | <eye_to_chin_profile_distance> | % 'three digit number' - unit= mm |
| LCP: | <lung_capacity> | % unit = litres |
| LFR: | <lung_flow_rate> | % unit = litres/minute |
| %---------- SPEECH/LANGUAGE | | |
| NAL: | <native_language> | % "French", " Spanish", etc |
| SOC: | <sociolect_of_target> | % "general", "broad", "cultivated",etc |
| ACC: | <strength_of_accent> | % (wrt) filename or "none" |
| INT: | <intelligibility_level> | % "high" or 'filename'.int = ASCII desc. |
| FLU: | <fluency_of_speech> | % fluent, moderate, disfluent |
| HER: | <hearing> | % "good", "moderate", "poor" only |
| SPQ: | <speech_quality> | % "normal", "lisp", etc |
| %---------- FAMILY INFORMATION | | |
| MNL: | <mothers_native_language> | % "German", "Russian", etc |
| MOC: | <mothers_occupation> | % "Baker", "Engineer", etc |
| MPO: | <mothers_place_of_origin> | % 'city/country' |
| FNL: | <fathers_native_language> | % "German", "Russian", etc |
| FOC: | <fathers_occupation> | % "Baker", "Engineer", etc |
| FPO: | <fathers_place_of_origin> | % 'city/country' |

**Table 2.** NSLD descriptors for static speaker characteristics.

| | | |
|---|---|---|
| SID: | <speaker_identifier> | |
| %---------- SPEAKER HISTORY RECORD | | |
| Cnn: | <category> | % 11 categories see below |
| Ynn: | <type> | % sub-category appropriate to category, |
| Lnn: | <level> | % hours per week or level '1..3' |
| Bnn: | <start_year> | % first year of episode |
| Enn: | <end_year> | % last year of episode |
| %---------- ELEVEN CATEGORIES | | |
| % language, singing, voice training, vocal stress, education, | | |
| % residence, music instrument, smoking, medical, sport, employment | | |
| %---------- TYPE EXAMPLES | | |
| % language: | | greek, italian etc. |
| % singing: | | opera, rock, professional, amateur, etc |
| % voice training: | | purpose |
| % vocal stress: | | teaching, noise in workplace, etc. |
| % education: | | primary, tertiary etc. |
| % residence: | | country, city |
| % music instrument: | | flute, clarinet etc. |
| % smoking: | | cigarettes, pipe |
| % medical: | | voice related condition, operation |
| % sport:netball, cricket, tennis etc. | | |
| % employment: | | teaching, student etc. |
| %---------- LEVEL EXAMPLES | | |
| % language: | | 1 = not fluent, 2 = moderately fluent, 3 = fluent |
| % education: | | year three, etc. |
| % residence: | | n/a |
| % smoking: | | number of cigarettes per week or hours per week |
| % medical: | | severity of condition |
| % other categories: | | hours per week |
| % ---------- BEGIN / END YEAR | | |
| % if Bnn: > 1000 and Enn: > 1000 then value = four digit year | | |
| % if Bnn: = 0 or blank and Enn: < 1000 then Enn: = number of years | | |

**Table 3.** SPEAKER_HISTORY descriptors in RDBMS.

It should be noted that the relational scheme necessitated two independent tables for SPEAKER and SPEAKER_HISTORY where the former contained all data that described their status at the time of recording as illustrated in table 2, whereas the latter contained all data referring to relevant episodes within their life as illustrated in table 3. Each episode was modelled using 5 variables but the number of such episodes was highly variable. Using both tables linked by the speaker identifier (SID) column

in each, current speaker characteristics and the speaker's prior speaking experience could, for instance, be used for selection.

### 4.1 SQL Queries

Queries in the RDBMS system using standard query language (SQL) can be written to access one or more of the relational tables. Here we will illustrate queries using a single table as well as using four tables to extract data files with specific characteristics relating to their speaker's characteristics, their speakers' history, the material spoken and the quality of the signal.

```
SQL> run
  1  SELECT sid
  2  FROM speaker_history
  3  WHERE cnn like 'smok%' AND ynn like '%igarett%'
  4* GROUP BY sid HAVING SUM(to_number(enn) - to_number(bnn)) > 5
RESULT> SID: s111, s126, s128
```

**Query 1:** *List the speakers who have smoked cigarettes for a cumulative period of more than 5 years.*

```
SQL> run
  1  select distinct speaker.sid,
  2  from speaker,speaker_history,ddf,quality
  3  where speaker.sid=speaker_history.sid
  4    AND speaker.sid = ddf.sid
  5    AND ddf.dfn = quality.quanum
  6  AND ddf.pid = 's017'                    /* sentence 17 */
  7  AND quality.npl='0' AND quality.nnl='0' /* no clipping */
  8  AND to_number(quality.med)>1800         /* high energy */
  9  AND to_number(speaker.yob) > 1970       /* born after 1970 */
 10  AND speaker_history.cnn like 'mus%'     /* musical_instr. */
 11* AND speaker_history.ynn like 'piano%'   /* sub-cat: piano */
RESULT> SID: s115
```

**Query 2:** *List all speakers who generate a high energy version of sentence 17 but without any signal clipping and who were born after 1970 and who play the piano.*

With ORACLE running on a dual-processor Sparc-10 system Query 1 utilised 1.17 cpu-seconds while Query 2 took 9.30 cpu-seconds. However, these figures were measured while restricting the search to only the YB_Sentences data. Applying the same queries to the entire ANDOSL data set yielded significantly better results: 0.14 and 7.53 cpu-seconds, respectively.

## 5. ANDOSL IN AN OO-DBMS FRAMEWORK

The structured descriptive files existing in ANDOSL contain a diverse variety of real-world attributes concerning speakers, recording conditions, file locations, etc. However, details regarding these real-world objects may be scattered among several files creating inherent difficulties in conceptualising the scope of the data, discriminating between variant and invariant information, and determining object relationships. For example, an unnatural dichotomy occurs between *speaker* and *speaker_history* since episodic information is a complex data type having anywhere from two to five values and is not suitable to be represented in the same RDBMS table with other one dimensional speaker data. This is an example where the representational limitations of the attribute-based RDBMS scheme force data to be partitioned in a contrived manner.

For these reasons an entity-based object model of ANDOSL was created that would correspond to the real world, simplify the expression of queries, and facilitate rapid DBMS response. The following aspects were observed during its genesis:

i) **Abstraction** A real-world model was first drawn out without regard to detail that emphasised the natural relationships between objects. For example, since recordings took place within an anechoic-chamber, a *recording-space* object was defined as well as other objects such as *microphone*, *reflective-surface*, speaking *task*, and *speaker*. The signal path from the *speaker* through *microphone*, signal *channel*, signal *transform*, *primary-representation* file, and finally to *recording* was made explicit via hard-wired relationships.

ii) **Encapsulation** Since external specification was separated from internal implementation, objects were given functional interface methods that facilitated the writing of queries. For example, a database object *x* could be asked "what speakers do you include" by a call to the generic function *speakers*, as in (*speakers x*), and would return its speaker objects in a list. Likewise, a microphone *x* could be asked to supply all of the speakers who have ever spoken into it by calling (*speakers x*), or all of its recordings by (*recordings x*).

iii) **Modularity** Organising ANDOSL data into groups of closely related objects promoted coherence and understandability. Scaling up the design to handle the MAP tasks with dual-channel recordings would not require a redesign but rather the addition of some new relations only. By shifting from an attribute-based model with 154 parameters to an entity-based model the system became more tractable since only 21 different object classes existed.

One possible object-oriented model that was implemented for ANDOSL data can be seen in figure 1. Data was read in from the CDROM files and distributed according to this set of object classes. A value-based identity was chosen to ensure data integrity, i.e., only uniquely valued objects were entered into the network. The system determined semi-automatically the minimum explicit relation types (indicated by arrows) required between objects, i.e., either a 1-to-1 or 1-to-many relation, given the set of desired relationships that were chosen a priori. Logical real-world modelling was focused on, e.g., in theory a primary-representation (the original recording) may have spawned several recordings even though this was not the case with these data. This object oriented division of ANDOSL YB_Sentence material formed a densely linked network structure with 7447 unique objects and 66944 links (1-to-1: 34096; 1-to-many: 32848).

Figure 1 can also be seen as a roadmap for the user to navigate with while forming queries. Traversal between objects not directly linked to one another is supported via computational links that are available for use, e.g., a speaker *x* can gain access to all of his/her recording objects (200 in this case) via the function (*recordings x*) which returns them as a list.

### 5.1 QuickSig Queries

Identical queries to those specified in section 4.1 were written as predicate functions in Lisp/CLOS, the implementation language of QuickSig, and applied to all speaker objects of the OODB. In the following lambda expressions (unnamed functions) the variable *speaker* refers to the object being tested. In the first query a speaker is first requested to return all cigarette smoking episodes as a list of episode objects via a call to *find-episodes-for*. Each episode then calculates it duration and these values are summed to obtain a cumulative value. If the value is larger than 5 years then the predicate returns *t*, otherwise *nil* (Lisps true and false values, respectively). Given this query the OO-DBMS returned the actual speaker objects as a list matching the predicate within the network structure. Further processing can be applied immediately to these objects if desired. As an indication of system performance database access time was 1.6 ms and 512 bytes of temporary memory was utilised which was recovered automatically by Lisp's ephemeral garbage collector. In the second example several conditions need to hold for the predicate

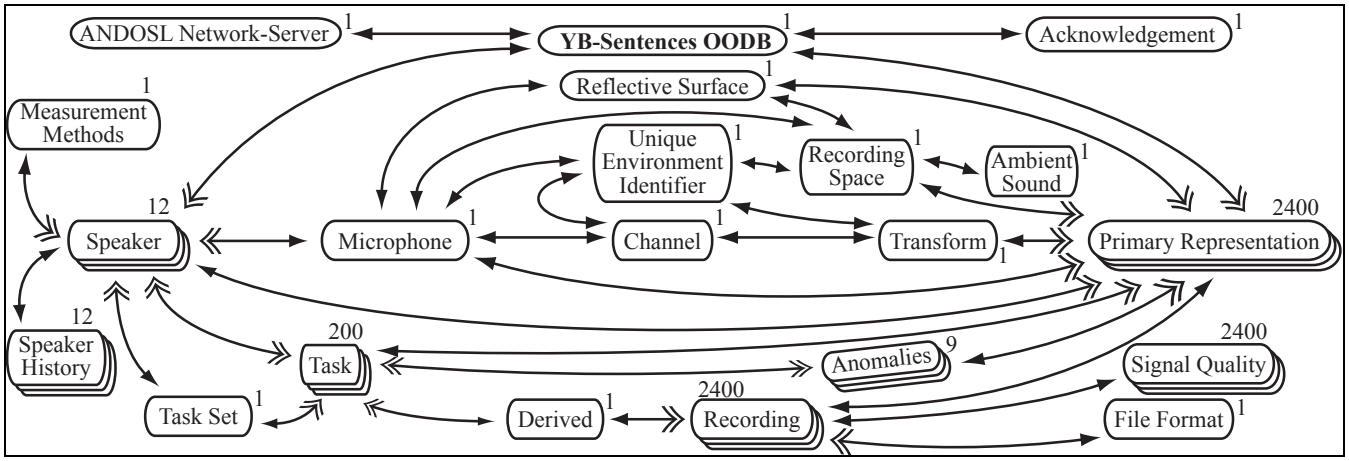


**Figure 1.** An object-oriented model of ANDOSL YB_Sentences built around 21 different classes. 7447 unique objects exist in this space (number next to class indicates how many instances per class occurred). Lines indicate direct relationships: 34096 1-to-1 links (single arrow) and 32848 1-to-many links (double arrow) create a dense structure on which database access occurs by traversal of links. Only nine anomaly objects were found since four speakers share the same object. This structure with phonemic data occupied 7,5 Mb of memory.

to return a non-nil value and this behaviour is obtained via Lisp's *and* operator: *and* evaluates each *form* sequentially. If *and* reaches a *form* that returns nil, it returns nil without evaluating any more forms. If it reaches the last *form*, it returns that form's value. The speakers year-of-birth is first tested to be larger than 1970. Then a test is made for at least one episode to exist for speaker where music and piano have existed. Finally, an iteration over all of the speakers primary-representations via a call to (*primary-representations x*) which returns them in a list. Then access to the primary-representations recording and its signal-quality object (see links in figure 1). When the remaining criteria are fulfilled then *loop* returns *t* and terminates immediately, else after looping over all *primary-representations* and failing to find a match *nil* is returned instead. This query returned speaker S115 as an object in 4.1 ms while allocating 40 bytes of memory. Tests were run on a 300 MHz PowerPC 750 RISC CPU with all structure in RAM running Macintosh Common Lisp 4.3b1.

```
(lambda (speaker)
  (let* ((episodes (find-episodes-for speaker "smok" "igarette"))
         (cigarette-durations (duration-of-episodes episodes))
         (cumulative-years (sum-of-elements cigarette-durations)))
    (> cumulative-years 5))))
=> (#<SPEAKER "S111"> #<SPEAKER "S126"> #<SPEAKER "S128">)
```

**Query 1:** Expression in OODB formalism (see section 4.1).

```
(lambda (speaker)
  (and (> (year_of_birth speaker) 1970)
       (episode-exists-for speaker "mus" "piano")
       (loop for primary-representation in (primary-representations speaker)
             for recording = (recording primary-representation)
             for signal-quality = (signal-quality recording)
             do (when (and (= 0 (number_of_positive_limits signal-quality))
                           (= 0 (number_of_negative_limits signal-quality))
                           (search "s017." (data_file_name recording)))
                  (return t))))))
=> (#<SPEAKER "S115">)
```

**Query 2:** Expression in OODB formalism (see section 4.1).

Queries are not interpreted during database access: Lisp's compiler generates a function from the query which is repeatedly applied by the search engine to seek out matches. Compilation time is included in the above performance figures.

## CONCLUSION

This paper has discussed some of the basic differences between an RDBMS and an OODB representation of a subset of speech data description. It has illustrated some of these differences by applying them in an entirely parallel way to the same speech representation tasks. The reader can, herein, examine the complexity of equivalent queries. Further work is required to more formally compare such facets as computational resources and scalability, both of which will be of significance as larger corpora are described to the extent indicated herein.

Both RDBMS and OO-DBMS systems are viable solutions for representing speech data. Selecting which database paradigm is more suitable depends upon many factors such as DBMS technology available and end-user requirements. An object-oriented view of speech data may be desirable in cases where further signal/linguistic processing is necessary since often a more suitable impedance match exists between the application and data: in OODBs the programming language of the application and the query language of the database can be identical and therefore seamlessly integrated. In such cases searches can readily return objects within the representation structure facilitating further processing, such as is the case within QuickSig. In the examples presented in this paper the RDBMS queries were written in SQL, a well known and formalised database query language. In the OO-DBMS system designed and constructed for ANDOSL data, Lisp was used which in this case forces the user to be proficient in Lisp to some degree even though a library of frequent queries can be formed. Finally, intuitive routes through the descriptive space will be tortuous unless flexible and natural interrogative structures are made available for the speech researcher to utilise.

## REFERENCES


[1] Millar, J.B., Vonwiller, J.P., Harrington, J.M., Dermody, P.J. (1994) "The Australian National Database Of Spoken Language", Proc. ICASSP-94, Adelaide, 19-22 April, Vol.1, pp.97-100.

[2] Millar, J.B. (1992) "The Description of Spoken Language", Proc. SST-92, Brisbane, 1-3 December, pp.80-85.

[3] Karjalainen M., Altosaar T., Alku P., QuickSig - An Object-Oriented Signal Processing Environment. Proc. IEEE ICASSP-88, NY 1988.

[4] Stajano, F., "A Gentle Introduction to Relational and Object Oriented Databases". ORL Technical Report TR-98-2. May, 1998. (http://www.uk.research.att.com/~fms/db/)

[5] Blaha, M., Premerlani, W., in *Object-Oriented Modeling and Design for Database Applications.*ISBN 0-13-123829-9. Prentice-Hall, 1998.

[6] Vonwiller, J., Rogers, I., Cleirigh, C., Lewis, W. (1996) "Speaker and Material selection for the Australian National Database of Spoken Language", Journal of Quantitative Linguistics Vol. 2:3, pp.177-211.

[7] Millar, J.B., Harrington, J.M., Vonwiller, J.P. (1997) "Spoken Language Resources for Australian Speech Technology", Journal of Electrical and Electronic Engineering Australia, Vol.17:1, pp.13-23.

[8] Harrington, J., Cassidy, S., Fletcher, J., McVeigh, A., The mu+ system for corpus based speech research, Computer Speech & Language, Vol. 7:4, pp. 305-331. October 1993.

[9] Millar, J.B. (1998) "A structure for comprehensive spoken language description", Proc ICLRE, Granada, 28-30 May, pp.1303-1308.