A Multimodal Database Framework for Multimedia Meeting Annotations

Hassina Bounif, Oleksandr Drutskyy, Fabrice Jouanot, Stefano Spaccapietra
Swiss Federal Institute of Technology, Database Laboratory
FirstName.LastName@epfl.ch

Abstract

The main objective of this paper is to present a flexible annotation management framework for a multimedia database system, applied to meeting recordings. Presented research and development activities are carried out within the scope of the IM2 project in which annotations play an important role in describing raw data from various points of view and in enhancing the query process. We focus on a database system capable of managing annotations (e.g. text transcriptions, dialog acts, speaker space position, etc.) and keeping links with raw data (audio, video, digital documents). This database provides a schema evolution mechanism and a meta-description layer ensuring flexible and incremental annotation definitions. To enhance this database system, some research works are currently in progress: a predictive methodology for schema evolution and a query technique that deals with fuzzy concepts and ontological commitments. We describe our on-going prototype development, in which we focus on data storage and interactive data access.

1. Introduction

Complexity of multimedia data requires sophisticated processing capabilities. Indeed, an audio stream, a video or a simple picture contain many different information. In order to provide an intelligent access to these data, it is required to add some metadata to multimedia documents. These metadata represent semantic descriptions produced during an annotation process, with each annotation characterizing some semantic dimension of the media. Also the users can change the dimension and modality of their interest while accessing the data. In this paper we are interested in proposing a solution to optimize meeting and annotation storage and to enhance querying and browsing capabilities for multimedia and multimodal applications.

This work is part of the IM2 Swiss national project[1] aiming at developing innovative solutions for interactive and multimodal access to multimedia data. Topics addressed include speech recognition, scene analysis, dialog understanding and indexing. A smart meetings application serves as a test case to demonstrate the results of the project. The application consists in storing and retrieving data from a series of project meetings. The data includes multiple audio and video recordings of each meeting, as well as structured and unstructured multimedia documents related to the meetings. All these raw data are annotated according to multiple criteria (e.g., speaker, topic, dialog act) to provide for rich information retrieval functionality. Supporting data management functionality addresses three main objectives. The first one is loading into a database system the multimedia data, metadata, and annotations coming from smart meeting rooms (rooms specialized for meeting recordings) and produced during the annotation process by different research teams (speech recognition, speech transcription, video analysis, dialog acts analysis, document analysis, etc.). The second objective is the availability of a powerful interface to query and browse meeting data using semantic information defined in metadata and annotations. Finally, visualization of all types of data and query results using a multimodal and interactive approach is the third objective.

Although nowadays a big number of research efforts in the domain of multimedia content annotation is based on the emerging MPEG-7 standard [9], which offers comprehensive audiovisual description possibilities and addresses a broad range of multimedia applications and requirements, this standard still has a number of limitations. Thus, one of the principal advantages of MPEG-7 and at the same time its shortcoming is its primarily orientation towards the physical, not conceptual model. Indeed, designed as a standard for description of AV sources, MPEG-7 only provides an XML Schema representation corresponding to the physical model, and understanding the semantics of MPEG-7 is sometimes difficult. Another major inconvenience is a lack of powerful publicly available software tools to work with the format, which is obviously partially related to its immaturity.

1 Interactive Multimodal Information Management, supported by the Swiss National Center of Competence in Research (NCCR), http://www.im2.ch

In the context described above we are developing an application framework based on a relational database enhanced by a meta-dictionary to deal with annotation semantics, a set of tools to manage annotations and meeting data, and a user-friendly query interface relying on a multimodal visual approach. The database contains knowledge of media files stored on a media file server. Adopting this concept, many heterogeneity problems have to be taken into account. Meeting recordings come from different meeting rooms, have different structures and different media formats (different number of cameras and microphones, different source channels). Thus it is important to allow enriching the database with all possibly known meta-information in order to avoid semantic losses. The main problem to deal with is annotations. They are the base of rich data retrieval and at the same time the source of heterogeneity problems. Annotations are provided by different teams involved in the project in form of XML and DTD files. We have to deal with both structural and semantic data conflicts [10, 6]. Understanding data in an XML annotation file is required in order to add information to the database and to link it to the information that is already there. For example, if a file describes utterances, it is necessary to find the right meeting, right participants, the annotated document (a transcription file), and perhaps the topics related to these utterances. Structural conflicts can be partially resolved by using a DTD or an XML schema related to an annotation file. However to cope with semantic conflicts we need extra knowledge to guarantee a good understanding of an annotation description and the correct loading of the annotation data into the database. Another problem related to metadata and annotations is certainly the evolution of user and project needs, which implies evolution of the database schema. Metadata related to meeting recordings and annotations may change over time, and we have to consider an original schema evolution technique to deal with this problem. The last important problem is the querying capabilities of the system, since they are essential to facilitate user interactions.

In this paper we describe different aspects of our proposition of multimodal system dedicated to meetings and annotation management. A global architecture of the system, described in section 2, has been defined to support all project requirements for annotation management, i.e. meeting definition, annotation definition, data querying and multimodal visualization. We give a summarized overview of the prototype under development used to manage meeting data. In section 3 we present our ongoing research on annotation loading and data storage mechanisms. Here we define an intelligent translation from XML representation of an annotation to flat relational format. A schema evolution methodology based on a predictive mechanism promises good results on database schema evolution according to existing and future requirements. In section 4 we also present some ideas on a querying approach, which uses semantic expressiveness of user profiles and domain ontology. This query technique is able to resolve ambiguous and fuzzy queries using extra knowledge. The last section concludes our current state in research and development, and suggests some directions for future work.

2. A Multimodal Framework Overview

2.1. Architecture Overview

The global architecture of our multimodal database system framework is presented in figure 1. The four main components are: the multimodal database, the annotation manager, the query module and the visualization tool.

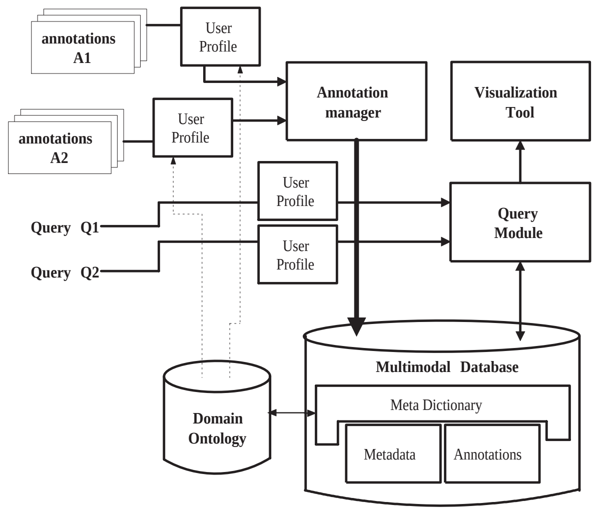


Figure 1. The Multimodal System Framework.

2.1.1. The Multimodal Database is composed of three layers:

- A metadata schema, which is the core of the DB schema. It defines the static information that a user can input to describe a meeting without a need for a specific analysis process.
- An annotation level, which contains the description of annotations of all types, using a structure optimized for data-storage.
- A meta-dictionary, used to enhance the semantics of metadata and annotation descriptions. For example, semantics of annotations and links between different types of annotations are clarified.

An annotation generally contains a lot of information and has a complex data structure. This means that each annotation has a specific conceptual schema describing its data organization. This schema has to be transformed into a pure relational schema in order to be integrated into the global database schema. The goal of the meta-dictionary level is to clarify some semantic information, which will simplify database management and will add some semantics to the relational model. The meta-dictionary (see figure 2) is composed of several components, each one depicting some specific meta-information on an annotation type.

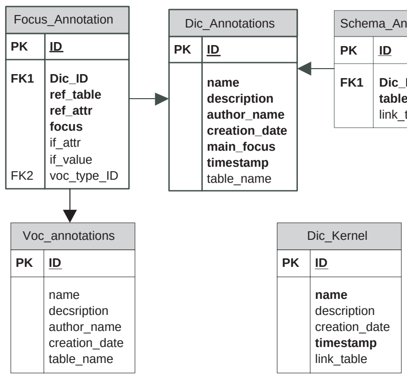


Figure 2. The meta-Dictionary.

The *Dic_Annotations* table is the main component, which lists the collection of annotation types defined in the database. It provides classical information about an annotation, like name, textual description, author, and creation date (for versioning). Some more important information appears as a *timestamp* flag saying if this annotation is timestamped or not, a *table_name*, which gives the name of the relational table dedicated to this annotation type, and a *main_focus*, which defines the main data type annotated by this annotation (e.g., utterances annotate participants, topics annotate meeting). This last information is useful to define an important link between an annotation type and other data that could be contained in a table from the core schema or in another annotation type. In the same way the *Focus_Annotation* component shows which annotation attributes contain links to other objects, i.e. data from other tables or annotations. It defines the following expression: ”the attribute *ref_attr* of the table *ref_table* contains links to the object *focus* iff the attribute *if_attr* of the table *ref_table* has the value *if_value*”. It also says if these links specify an entry in the ontology (*voc_type_ID*, vocabulary for annotation if defined in *Voc_Annotations*). The *Schema_Annotation* component defines the relational schema of an annotation type, i.e. the list of tables involved for its representation, and indicates if a table is just a structural link table or an information table. The *Dic_Kernel* defines the core database schema.

Figure 3 gives an example of defining an annotation on speaker utterances in the database using the meta-dictionary structure. This is a small snapshot of the database schema, which shows the main meta-dictionary table *Dic_Annotations*, four main tables of the core schema (*Meeting*, *Person*, *Participant*, *Document*), and one annotation table (*Utterance*). An entry is created in *Dic_Annotations* to define this annotation as following: $\langle$ ”Speaker Utterance”, ”Utterances which compose a turn of speech for a participant of a meeting”, ”LBD team”, 07/14/2003, ”Participant”, true, ”Utterance” $\rangle$. The *Utterance* is built from a generic annotation template composed of three attributes: an *Id* to identify each annotation instance as an object, a *Data_ID* referencing the focal object (i.e. annotated information), and a couple (*stime*,*etime*) for (start time, end time) to store the time-stamped data. The *Data_Id* is thus a foreign key, which represents an object of the *Participant* table. The attribute *text* contains the text transcription of the dialog (specific information for this annotation). *Transcription_ID* represents an object from the table *Document*, i.e. the file that contains the global transcription of a meeting. It is also defined as an entry in the table *focus_Annotations*. And for this particular annotation, *scar* and *ecar* store the position of each utterance in the global transcription file. Thus, the meta-dictionary structure allows modeling a part of the semantics in the utterance annotation, mainly the objects that are linked to the utterances.

We aim at defining a domain ontology [4] to give a clear definition of all the concepts used in different domains of the IM2 project and to enhance the *Voc_Annotations* structure. At the current stage a set of concepts can already be formally defined. However, most of them will be introduced as updates to the ontology as new information and new annotations arrive. Ontology at this level consists of the domain vocabulary, and a concept is a taxonomic definition having its name extended by synonyms, roles and links with other concepts. User profiles, based on this ontology, help to contextualize the domain. They are used to assure a good understanding of user needs, and are important at two levels: optimizing the understanding of user annotations, and query enhancement.

2.1.2. The Annotation Manager This module plays two important roles in our multimodal system. The first one is definition of annotations, i.e. the possibility to add, to modify, to delete an annotation type, and to populate it with annotation values. The second role concerns the problematic of schema evolution in such a dynamic research environment as multimedia and multimodal data.

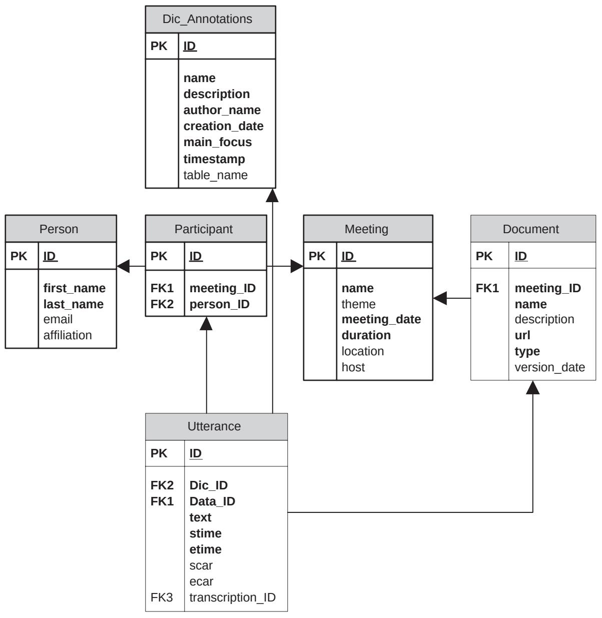


Figure 3. The representation of utterance annotation.

Because provided by people from different research domains, annotations are heterogeneous and it is important to understand the semantics of an annotation file. The annotation manager has some functions to define and understand an annotation. The first one is the addition of a new annotation using an intelligent translation mechanism that builds a database annotation schema from an XML annotation file. The structural translation from XML representation to database model (relational) is based on XSLT filter. In this approach an XSLT filter has to be written for each annotation type. We plan to use a CPI-like algorithm [7] enhanced by extra information represented in a user profile. This information could clarify the mapping between user annotation XML format and internal database format. The second important function is a controlled modification and deletion of annotations to ensure that the database and links between data keep coherent. For example, an annotation cannot be expunged if it is annotated by another existing annotation.

The previous problem of annotation deletion and modification is related to a more general problem, which can be summarized as a database evolution problem. Objectives of the annotation manager are to take into account the multimodal database evolution in the two following aspects: (1) a predictive evolution, which allows incremental evolution of the database conceptual schema by applying rules, (2) a temporal evolution, which keeps available different versions of the same annotation in the database. This predictive mechanism keeps the database in a coherent state without a constant need to come back to conceptual stage. This functionality is based on semantic information describing current and future information system and on mining tools.

2.1.3. The Query Module In our system we propose an ontology-driven technique that facilitates the query composition by guiding the user through the concepts present in the annotation database. Doing this we plan to take advantage of dynamic user profiles that help to better adapt the system to user needs and preferences. Thus, for example, if a user specifies an ambiguous concept (having several possible interpretations) in his request, the system could help him automatically select one particular interpretation based on observations of user's past interactions (i.e. selecting the interpretation that corresponds the most to the user's current profile). To facilitate the machine access to our database system we propose an approach based on web services.

2.1.4. The Visualization Tool The visualization tool is of particular importance for a multimedia and multimodal database system. It provides a complete framework to access data in a user-friendly manner. Because IM2 is research oriented, user needs evolve continually providing new data, new annotations, and new applications to make the most of them. Our goal is not to build a complex user interface integrating all functionalities, applications, and visualization tools required by users. On the contrary, we provide a flexible and open architecture allowing browsing and visualizing any type of annotations. We also provide the possibility of adding new specific browsing and visualization plug-ins.

The visualization tool is composed of two important components closely related to each other. A visualization engine can play Web contents in different windows to give a multidimensional view of a meeting (audio, video, pictures, documents, metadata and annotations). This engine ensures that all windows are synchronized with a source media. It provides a time server and a client API to manage time events thus simplifying visualization plug-in development. The other main component focuses on interactivity and browsing capabilities integrated in the visualization engine. Thus the entire window content can provide some user interactivity. The intelligent query module will also be embedded into this tool to enhance querying capabilities.

2.2. A Prototype Application for Meeting Analysis

In this section we present a working prototype for IM2 meeting management. Although further developments are still necessary and many features of annotation management are not yet implemented, it is an interesting example of a multimodal system dealing with user requirements. The working prototype is composed of five important Web modules with a central component being an Oracle database.

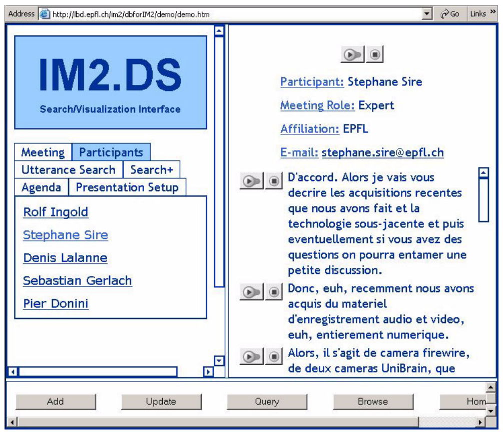
(1) The database schema with its core part and its annotation part is completely depicted in [1]. The core part of the schema defines all relevant information about a meeting: persons involved in it, participants (speakers), related documents (slides, articles, books, etc.) with their authors, topics, schedules (classical one or minutes). Obviously a meeting is associated with a set of audio and video files composing multimedia sources of a meeting recording. Currently a collection of annotations is defined in the database: turn of speech for participants, topic events, utterance text transcription for participants, logical media streams. This last one allows defining a virtual audio/video stream from several streams (audio and video files). It is thus possible to follow a participant filmed by different cameras, to highlight a specific participant if many people are present on the same video shot, and, in general, to edit a virtual stream. All these annotations are specified using the meta-dictionary structure. Many new annotations are to be added soon, including but not limited to dialog acts, document analysis, and participant spatial localization. All these annotations allow for complex queries like (a) *Which participant talks more than the others?*, (b) *In which meeting has this speaker shown this particular presentation (slide file)?*, (c) *Show me all the meeting samples where this speaker has spoken on this topic?*, and more.

(2) The meeting update interface is a classical Web-form interface, which provides all the features to enter basic data of a meeting and a set of metadata. Users can easily specify audio and video files of the meeting recording. They can also add topics of interest, participants, agenda, and minutes notes. Such information forms the base for future annotation definition and querying.

(3) The annotation update interface is not yet implemented (remains a manual process). This is an intermediate solution until the user XML annotation translator is completely formalized. Users can upload their XML files and we provide an XSLT filter to translate their data into a database-ready format.

(4) The browsing interface is a simple query interface providing all essential information about a meeting and allowing to playback a meeting also displaying some annotations. The interface provides some basic query capabilities using keyword search. This interface is currently utterance-oriented, but evolves to allow for more general browsing capabilities. A screenshot of this module is shown in figure 4.

(5) the querying interface enables a user to construct complex queries in a simple user-friendly way. Without knowing the database structure a user can express his requirements by simply selecting relevant concepts from the core database schema and annotations. He can complete his requirements by selecting objects, which are connected with the selected items. It works as a sort of a browsing system,



http://lbd.epfl.ch/im2/dbforIM2/demo/demo.htm

Participant: Stephane Sire

Meeting Role: Expert

Affiliation: EPFL

E-mail: stephane.sire@epfl.ch

D'accord. Alors je vais vous decrire les acquisitions recentes que nous avons fait et la technologie sous-jacente et puis eventuellement si vous avez des questions on pourra entamer une petite discussion.

Donc, euh, recemment nous avons acquis du materiel d'enregistrement audio et video, euh, entierement numerique.

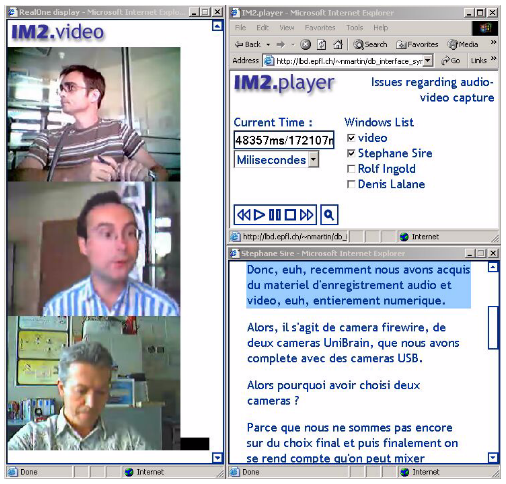
Alors, il s'agit de camera firewire, de deux cameras UniBrain, que

Figure 4. Browsing, querying and management interface.

where by choosing the information from a starting point it is possible to select information dependant on it, etc. All dimensions of information (basic concepts or annotations) can obviously be constrained. The results of a query can be obtained either as a downloadable file or visualized on the screen.

We are currently working on a visualization tool capable of playing SMIL [13] multimedia presentations (generally, a collection of video and audio streams representing a meeting recording session) that are synchronized with metadata. These metadata are usually annotations or results of queries (with timestamped results). The presentation is organized in a multi-window environment. Figure 5 shows a screenshot of our visualization tool: a meeting is played and it is possible to display the text transcription of participant utterances (participant windows scroll automatically to highlight the current utterance) in a separate window.

This visualization tool is based on a time server, which polls the RealOne SMIL session and sends time events to different client windows that visualize metadata. Client windows interact with the time server using an API that offers some simple services like create a new client, delete a client, play, stop, and seek. The client is based upon a default template that proposes a standard API to manage time events. The author of a client window has to focus only on what he wants to display and the way he wants to display it. This is a very flexible approach, which is not limited by the number of client windows, the type of displayed information and the manner of displaying the information.


RealOne display - Microsoft Internet Explo
IM2.video
IM2.player - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://lbd.epfl.ch/~nmartin/db_interface_syr
Go Links
IM2.player
Internet
Stephane Sire - Microsoft Internet Explorer
Done
Internet

Issues regarding audio-video capture

Current Time : 48357ms/172107n

Milisecondes

Windows List
☑ video
☑ Stephane Sire
☐ Rolf Ingold
☐ Denis Lalane

http://lbd.epfl.ch/~nmartin/db_i

Donc, euh, recemment nous avons acquis du materiel d'enregistrement audio et video, euh, entierement numerique.

Alors, il s'agit de camera firewire, de deux cameras UniBrain, que nous avons complete avec des cameras USB.

Alors pourquoi avoir choisi deux cameras ?

Parce que nous ne sommes pas encore sur du choix final et puis finalement on se rend compte qu'on peut mixer

Figure 5. Visualization Tool.

3. Annotation Management For Multimodal Database

In this part of the paper we present our ongoing research that aims at enhancing annotation definition and annotation storage in the multimodal database. It relates to the problematic of a good database schema design in such a dynamic environment as annotation storage for multimodal applications. The first main problem is loading annotations into the database. This introduces two related difficulties: the definition of a relational sub-schema for each new annotation from an XML specification, and addition of new annotation values from XML files. An annotation translation method allows extending the database schema with information from XML annotation files. We also focus on an original schema evolution approach consisting in predicting database evolution phenomena and anticipating changes on them.

3.1. Annotation translation

Annotation translation consists of transferring data from XML documents to relational database [2] using XSLT filters, PERL programs and SQL*Loader (an Oracle utility for loading data from flat CSV files into one or more database tables). The global process is exposed in figure 6.

This transformation is not just a naive data translation from one format to another. It involves advanced processing on XML annotation files, based on a simplified version of CPI (Constraint Preserving Inlining) algorithm [7] to preserve both semantic and structural aspects, which are discovered during the transformation of hierarchical XML model. Our transformation algorithm (see figure 7) regroups three processing steps of the

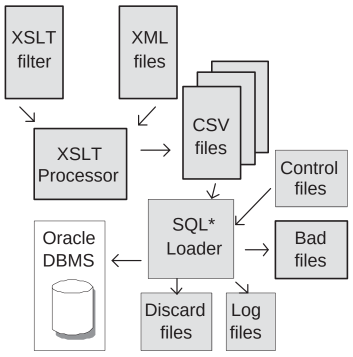


Figure 6. XML to database annotation translation.

CPI algorithm (i.e. transforming schema, discovering constraints via find constraints procedure, and preserving constraints via rewrite constraints procedure) into two steps: detection-transformation and CSV-files-generation.

```
Input  : XML annotation files
Output : CSV files (each CSV file corresponds to a database table)
Begin
 For every XML annotation file
 Begin
     Invoke XSLT filter for detection of constraints and transformation step
     Results are put in CSV-files
 End
 For every CSV file
 Begin
     Invoke Perl filters for generating primary keys
     Modify the other existing CSV files
     Invoke Perl filters for generating foreign keys
     Modify the other existing CSV files
 End
End
```

Figure 7. Transformation algorithm.

3.2. Schema Evolution for Multimodal Database

There is a large body of literature on database schema evolution. The proposed solutions usually rely on two principal approaches namely Schema Modification and Schema Versioning [8] [15].

Schema Modification:

Past states of the schema are not preserved. The old schema and its corresponding data are replaced by either one of the new schemata and its new data. The data associated with the schema are managed in two ways. In the first one, new data is created (the existing data is not considered). In the second way, existing data is transformed to be consistent with the new schema. Both ways may lead to information losses and incompatibility of applications using the old schema.

Schema Versioning:

The state of old schema and corresponding data are preserved. The entire schema changes are handled by creating different new versions of the schema. There are two different ways of versioning. In **parallel** versioning, the entire schema is perceived as a unity on which versions are created. In **historical** versioning, each version is independent of the others and is stored in a separate memory zone. Old versions are read-only. Therefore changes are made only on the current version. Traditionally, in this approach the number of versions is significant.

Unlike traditional approaches, we take a new look at the problem of schema evolution. We undertake an innovative research endeavor consisting in three main phases: pre-modeling, modeling, and evolution. Our approach is multidisciplinary. It involves such fields like database systems, ontologies, and data-mining techniques. Below we describe the details of each phase.

Pre-modeling: The goal of this phase is to define current and future user requirements. Therefore, for this step, no database conceptual schema is defined and in general, a database designer and system users are the main actors. The output is defined in terms of requirement ontologies. The pre-modeling phase is divided into two sub-stages. These are respectively acquisition and predictive stages.

In the Acquisition Stage, the database designer assisted by user interactions and feedbacks detects and defines the real requirements. He formulates these requirements and functionalities by using representation formalism such as ontologies.

In pre-modeling, the predictive stage is used to predict future user requirements and future tasks of the system. It is divided into two sub-stages. These are respectively implicit/similarity-driven changes and unexpected changes:

- *Implicit/Similarity-Driven Changes:* Our idea is to find out future requirements and functionalities of our system assuming an analogy of our system with those addressing the same problem domain. For example, consider the case of a digital music catalogues system. To express the expected or implicit changes that could happen to it, we study similar systems of other catalogues, e.g. online book catalogue systems.
- *Unexpected Changes:* Here, we will find out the unexpected future requirements. We use data mining techniques on dataset samples gathered in the previous step (Implicit/Similarity-Driven Changes). For example, in the case of a digital music catalogue we firstly collect enough web pages of book catalogues such as *amazon.com*. Secondly, we extract from these web pages keyword information to build datasets. Then, we apply data mining techniques on these dataset samples to find out these unexpected requirements for our system.

Modeling: In this phase, the main actor is the database designer and the inputs are the predictive module and the ontologies obtained in the pre-modeling phase. The predictive module is a set of programs (statistical computing program, search program, sort program, etc.), which takes the (current and future) requirements and functionalities formulated in the pre-modeling and automatically generates the conceptual database schema(s). These consist of a generic schema connected to one or more conceptual sub-schemas. The generic schema models the current requirements whereas sub-schemas model different future requirements and functionalities. The costs of conceptual schemas are calculated by the predictive module using statistical programs. And it is the database designer who analyses and chooses the conceptual schema among the proposed sub-schemas.

Evolution: This stage consists of using transformation mechanisms to evolve the current database schema. The main actor is still the database designer. We propose a modified versioning approach as shown in figure 8 rather than using the original one (a complete description in [15]). The reason are that: 1) our database schema has a pre-possibility for evolution: entities, associations, and attributes can be added, deleted or modified according to the database designer needs; 2) data provided by the database are in the form of metadata and modified data, instead of having several versions of data as it is the case in the original versioning approach. The metadata is essential to keep the historical evolution of data, its constraints, and its semantics.

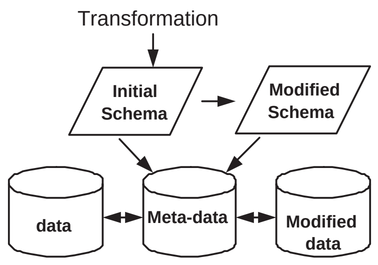


Figure 8. A modified versioning approach.

4. Multimedia and Multimodal Query Language

One of the most important features of a multimedia system is querying. In our approach we base ourselves upon accompanying metadata to provide querying possibilities. Thus, a user is able to use the meeting descriptive information (e.g. meeting place, persons involved in a meeting, etc.)

as well as the annotations (such as participant speech turns, documents presented, etc.) to formulate his queries to the system.

In our prototype system described in the section 2.2, we provide several querying possibilities with the most basic one being a keyword-based search. Although rather efficient in some cases, this technique proves to be very limited when it comes to semantic querying. That's why an ontology-driven alternative becomes essential for our multimodal annotation system.

4.1. Ontological Query Enhancement

Despite the relative simplicity, a keyword-based approach suffers from a number of shortcomings. Thus a keyword in a document does not necessarily mean that the document is relevant. On the other hand, a relevant document may not explicitly contain the given keyword. Synonyms lower recall rate, homonyms lower precision rate, and semantic relations such as hyponymy, meronymy, antonymy are not exploited. Keyword-based search is especially useful to a user who knows what keywords are used to index the information and therefore can easily formulate queries. This approach is problematic, however, when the user does not have a clear goal in mind, does not know what there is in the database, and what kind of semantic concepts are involved in the domain. To deal with the problems mentioned above, a number of ontology-enhanced approaches have been proposed (see [5], [11]).

In our approach we propose to use the domain ontology to help users exploit the semantics of the system. As it was mentioned in section 2, our ontology evolves over time as new annotation types are introduced. Since the IM2 project involves many research teams in various domains, the number of concepts managed by the system can grow drastically over the time. It is thus important to use a dynamic user profile mechanism able to extract only that part of the global ontology that would probably be of the most interest for a particular user. Since user preferences and points of interest tend to change, an updating mechanism that tracks user interactions with the system has to dynamically keep the user profiles up to date.

4.2. Fuzzy Querying Approach

In spite of all the advantages of ontology-enhanced techniques, in certain cases they can be too (undesirably) precise. That's why another important feature of the query subsystem being proposed is a possibility of constructing fuzzy requests. In [14] a fuzzy framework for manipulating temporal information in video information systems is proposed. In [3] the authors use the fuzzy sets theory to readjust the weights for ontological interrelationships. In our multimodal system a combination of ontology and fuzzy querying techniques enables a high level of system personalization and of coherency between the user request and the reply produced.

We distinguish fuzzy values (e.g. around 5 p.m.), fuzzy variables (e.g. by specifying a variable ”timestamp” the user could also mean a variable ”time” and/or ”date”, etc.), as well as fuzzy concepts (e.g. in the case of our system a concept ”turn” could mean ”speech turn” or ”document presentation turn”). Since it is clear that interpretation of fuzziness is to a big extent user-dependent, in our system we again benefit from user profiles to facilitate working with fuzziness depending on personal preferences of a user. Thus, for instance, ”around 5 p.m.” could mean ”5 p.m. $\pm$ 5 min.” for one user, and ”5 p.m. $\pm$ 15 min.” for another.

Let's illustrate a possible query scenario by the following example. A user wishes to obtain a speech transcription and an audiovisual presentation of a talk presented by a specialist in dialog act analysis in about 20 minutes from the beginning of the meeting. Constructing his query the user could use some visual or algebraic language, or a combination of both. Let's point out the main peculiarities of constructing such a request in our system. Firstly, to find out what concepts correspond to ”speech transcription” and ”audiovisual presentation”, the user browses the domain ontology and selects the appropriate terms. Secondly, finding out from his personal profile that the user is interested mostly in dialog act analysis, the system makes it possible for him to enter simply ”specialist” instead of ”specialist in dialog act analysis”, automatically disambiguating the entered term. Thirdly, based on the personal user preferences the system would personalize the fuzzy time value ”about 20 minutes”, converting it, for example, into ”20 min with probability 100%, 15 or 25 min with probability 75%, etc.”

4.3. Querying as a Web Service

Taking into account the diversity of users of our system, it becomes clear that providing a human-computer interface as the only means of interaction with the database does not suffice. Indeed, the variety of the research teams involved in the IM2 project makes it almost impossible to foresee all potential interaction scenarios between the users and the database system. Thus it is hardly feasible to make an interactive HCI that could satisfy everybody's requirements. Moreover, each research team involved usually has its own set of (client) software tools capable of querying information from external data sources. All this shows a strong need for providing automatic database access and querying possibilities for client software.

An obvious solution to this problem could consist, for example, in simply establishing a connection between the client application and the annotation database via a JDBC interface. However this approach suffers from a number of serious shortcomings, which in general can be characterized as follows: instead of concentrating on the (annotation) content, the client tool developer would rather have to focus on low-level schema and implementation issues. Obviously such an approach could hardly be called thoughtful and user friendly. That's why in our system we propose an approach based on web services [12]. In this way, the client uses a set of web services to query and manipulate the annotation database system content. As compared to the classical ODBC/JDBC-like approach, the proposed technique benefits from a number of advantages being:

- client software developer does not need to get much into details of low-level database implementation
- database migration, distribution, etc. become transparent for client tools (only the web services communicating with the database system would have to adapt to changes)
- database schema evolution/versioning also becomes transparent for the clients.

The last issue is of a big importance for us taking into account our research activities on schema evolution described in the section 3.

To suite best to user needs in no time, we propose to establish a set of pre-defined query functions accessible as web services. At the same time we are working on establishing a web service that will permit working with user-defined queries expressed in a freer form.

5. Conclusion and Future Work

In this paper we have presented our proposition for a multimodal system dedicated to annotation management in a meeting recording environment. Having in mind the requirements of the IM2 project that we are involved in, we have proposed a solution to the main tasks of annotation management, i.e. continuously adding and updating annotations, intelligent querying, interactive and multimodal visualization support. We propose a multimedia database system architecture that provides evolvability, flexibility and scalability. The core of this solution is a database built on an evolvable schema assisted by a meta-dictionary. The need to develop an XML translation technique and a predictive schema evolution approach to update incrementally the database has been shown. We also demonstrate the relevance of a query mechanism that uses concepts from ontologies and fuzzy logic theory. A working prototype is presented, however still a lot of work is required to produce a complete system with intelligent modules as depicted in this paper, and implemented as web services.

References

[1] H. Bounif, O. Drutskyy, F. Jouanot, and S. Spaccapietra. A flexible database schema for the im2 smart meeting application, 2003. Technical Report.

[2] R. Bourret, C. Bornhövd, and A. P. Buchmann. A generic load/extract utility for data transfer between xml documents and relational databases. In *Workshop on advanced Issues of EC and web-based Information Systems (WECWIS)*, 2000.

[3] V. Cross and C. R. Voss. Fuzzy queries and cross-language ontologies in multilingual document exploitation. In *Proceedings of the Ninth IEEE International Conference on Fuzzy Systems*, pages 641–646. San Antonio Texas, May 7-10 2000.

[4] N. Guarino. Semantic matching: Formal ontological distinctions for information organization, extraction, and integration. In . T. Pazienza, editor, *International Summer School (SCIE'97)*, pages 139–170. Frascati, Italy, Lecture Notes in Computer Science, Vol. 1299, Springer, 1997.

[5] E. Hyvönen, A. Styrman, and S. Saarela. Ontology-based image retrieval. In *Towards the semantic web and web services, Proceedings of XML Finland 2002 Conference*, pages 15–27. Helsinki, Finland, 2002.

[6] V. Kashyap and A. Sheth. Semantic and schematic similarities between objects in databases : A context-based approach. Technical Report TR-CD-95-001, LSDIS Laboratory, Department of Computer Science, University of Georgia, USA, 1995.

[7] D. Lee and W. W. Chu. CPI: Constraints-preserving inlining algorithm for mapping XML DTD to relational schema. *Data Knowledge Engineering*, 39(1):3–25, 2001.

[8] B. Lerner and A. Habermann. Beyond schemas evolution to database reorganisation. In *In Proceedings of ECOOP/OOPSLA 90 Conference*. Ottawa, Canada, October 1990.

[9] MPEG-7. http://www.mpeg-industry.com.

[10] A. Ouksel and A. Sheth. Semantic interoperability in global information systems. *ACM SIGMOD Record*, 28(1):5–12, March 1999.

[11] C. Tsinaraki, E. Fatourou, and S. Christodoulakis. An ontology-driven framework for the management of semantic metadata describing audiovisual information. In *The 15th Conference on Advanced Information Systems Engineering*. Velden, Austria, 2003.

[12] W3C. Web services activity. http://www.w3c.org/2002/ws/.

[13] W3C. Synchronized multimedia integration language, 2001. http://www.real.com.

[14] R. R. Yager. Retrieval from multimedia databases using fuzzy temporal concepts. In O. Pons, M. A. Vila, and J. Kacprzyk, editors, *Knowledge Management in Fuzzy Databases*, pages 261–274. Springer-Verlag, 2000.

[15] S. Zdonik. Object-oriented type evolution. In A. Press, editor, *In Advances in Database in Database Programming Language*. New York, 1990.