

# A Semantic Framework for Meeting Data Retrieval

Weisheng He  
Tsinghua University  
FIT Building, Tsinghua University  
Beijing, RPC

Peifeng Xiang  
Tsinghua University  
FIT Building, Tsinghua University  
Beijing PRC

Yuanchun Shi  
Tsinghua University  
FIT Building, Tsinghua University  
Beijing PRC

hws99@mails.tsinghua.edu.cn

xpf97@mails.tsinghua.edu.cn

shiyu@tsinghua.edu.cn

## ABSTRACT

Meetings are often captured for future access in daily life and with an increasing amount of such record, computers should be used to process and retrieve the meeting content to improve the access efficiency. This requires a semantic framework to formally represent and model the meeting procedure and user intentions so that they can be processed by machines. In this paper, we present a framework, MiF (Meeting information Framework), to structurally formalize meeting contents and user retrieval queries. MiF consists of four components: the primitive ontology provides the schema to define basic semantic units and their interrelation; the archive expression model (AEM) enables a comprehensive representation of the meeting procedure; the query expression model (QEM) supports the formalization of high-level, semantic-rich retrieval queries; the query processor executes queries and makes necessary inference based on the primitive ontology. With this framework, a prototype system for meeting record has been implemented, by which hypermedia data can be automatically captured and complex user queries can be expressed and processed for an efficient retrieval.

## Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval – *Query formulation*; H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval – *Retrieval Models*;

## General Terms

Management, Design, Human Factors

## Keywords

meeting record, formalization, ontology, semantic primitive, inference engine

## 1. INTRODUCTION

Meetings are important activities for team communication and coordination, generating a large amount of information which is not formally documented. Thus the utilization of computers to

automatically capture meetings as compound multimedia records (including video, audio, text etc.) has become a research topic for more than a decade to ensure the rich produced information during meetings will be available for future access.

For post-meeting access, few users are willing to sequentially navigate the records since they will have to spend hours on many unimportant, boring segments before reaching the information they really want. Instead, users prefer to directly locate and jump to the interesting points to save time and energy. Abowd et al. [1] have reported that in the authentic use of eClass system in GaTech, over 85% of the access to lecture records is not sequential. In this sense, a key factor to the success of meeting capture and access systems is to provide meaningful indices with rich semantic to enable users to quickly locate the information they want and index into other related media for further details.

There have been considerable efforts[2][3][4][5][6][12][14] towards this direction by detecting various user behaviors and important events and keeping them as record indices. The semantic of these indices compose a structured representation of the meeting procedure (in contrast with the stream media such as video/audio which provide no structural information that are machine-processable). However, due to the large amount of records that will accumulate after long term use, there is still an imbalance between the volume of structured representations and the human processing capability. This situation calls for the machines to bridge the gap by processing and retrieving the information instead of humans.

Before the meeting data can be efficiently processed and retrieved by machines, a unified framework should be established to formalize the rich semantic information in meetings data and user queries due to the following reasons:

1. To enable machines to understand and process the multifarious queries and relate them to the semantic of meeting procedures, a model is required to unify their representations so that they will be comparable.
2. To avoid redundancy, much information is not stated but can be implied. For instance, if the following knowledge are explicitly declared in a meeting record and the metadata: 1) Eric hosts the meeting; 2) Eric is a senior manager; 3) Any meeting hosted by a senior manager is a top-level meeting, then from these facts a conclusion can be drawn that this meeting is top-level and should be carefully reviewed. For machines to understand these implications and make correct reasoning, there should be a model for formalization and inference rules.

In this paper, we present a semantic framework MiF (Meeting information Framework) to formalize the information relating to meeting capture and access. The motivation of MiF is to formally represent and model the rich semantics during meeting

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CARPE '05, November 11, 2005, Singapore.

Copyright 2005 ACM 1-59593-246-1/05/0011...\$5.00.

procedures which can be captured by various sensors so that the records can be efficiently processed and retrieved by machines. It is a research theme under a large umbrella project SEMIC<sup>1</sup> [7] focusing on the building of a smart space to enhance the user experience in collaborative activities.

Previous research[2][8][9] on meeting modeling only focus on the representation of the meeting procedure and little is addressed on how users can make their need for information understood by machines through formalized queries. We argue this inadequacy is a result of the lack of a unified framework to model both the semantic of meeting data and user queries. Thus, MiF differs from these previous works by further providing a model to formalize user queries with rich and complex semantic and interrelations, as well as the capability to process such queries. The MiF is composed of four components: primitive ontology, archive expression model (AEM), query expression model (QEM) and a query processor. MiF is a general solution because only the primitive ontology is related to application-level semantic (e.g., the user name and roles, the possible action type and certain reasoning rules) of meeting scenarios. Thus the implementation of the other three components can be reused in multifarious meeting scenarios and only the primitive ontology needs to be reconfigured in case of a scenario change. Although MiF is primarily designed for usage in meetings, it can also be extended to model other collaborative activities such as those in classroom or in command posts.

The rest of this paper is organized as follows: section 2 introduce details of the four components of MiF; in section 3 we describe a prototype meeting capture and access system implemented on MiF and the binding techniques employed; Section 4 concludes the paper and outlines our future work.

## 2. The MiF

MiF is of a layered structure as shown in figure 1. The underlying primitive ontology defines the basic semantic units composing the meeting information space and their interrelations for high-level information representation and processing, while AEM and QEM are used to formalize the structured representation of meeting procedures and user queries, respectively. Based on these three components, the query processor will execute the retrieval and make necessary inferences with its core component, the inference engine. Details of these components will be discussed in the rest of this section.

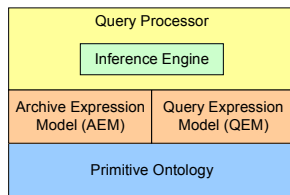


Figure 1. Layered structure of MiF

### 2.1 The Primitive Ontology

Any information in a meeting procedure is a combination of certain semantic units in a certain structure. These units are termed as primitive, which is the atomic and indivisible item to

convey information. For instance, to express information through a statement “Andy enters the meetingroom”, we use three primitives *Andy*, *Enter* and *Meetingroom*. Hence features and interrelation of these primitives should be the basis for further exploration towards the representation and modeling of semantics at higher levels. This underlying knowledge is defined as **primitive ontology**.

Ontology is generally defined as the “representation of a shared conceptualization of a particular domain”. There have been many forms to represent ontology and in MiF the taxonomic hierarchies of classes and subsumption relation is employed. As shown in figure2, all primitives can be classified into one of the four types arranged in a matrix. The abstraction of unary entities is denoted as **class**, which defines a group of individuals that belong together because they share some attributes, e.g., *Man* is a class denoting humans whose gender is male. The abstraction of binary relations is denoted as **property**, which describes the connection between individuals or from individual to data value, e.g., *FatherOf* is a property to state that the first individual is the father of the second individual. Class and property both have corresponding instances which are termed as **individual** and **property\_instance**. For example, *George HW Bush* and *George W. Bush* are two individuals of class *Man*, and the tuple (*George HW Bush*, *George W. Bush*) is an instance of the property *FatherOf*.

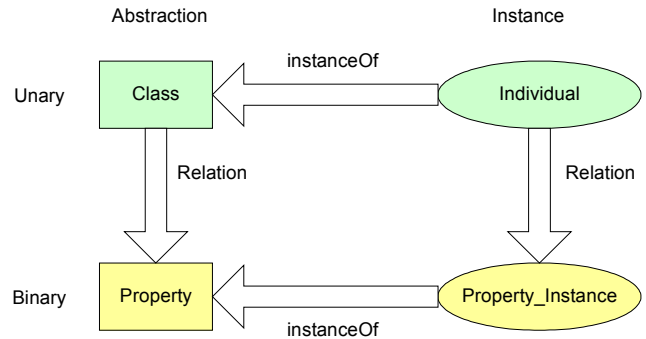


Figure 2. Matrix structure of primitive ontology

Class hierarchies can be created by making one or more statements that a class is a subclass of another class, e.g., class *Man* can be stated to be a subclass of class *Person*. From this an inference engine can deduce that if an individual is a *Man*, then it is also a *Person*. Property hierarchies can also be created in the same way. Similar to the relations in set theory, a property may also be declared as transitive, symmetric or the inverse of another property. These attributes are the basis for semantic reasoning to the meeting data.

For meeting procedures, we classify all semantics into five dimensions: time, location, user, action and object. Correspondingly, the primitive ontology can be divided into five sub-ontologies: *time\_primitive*, *location\_primitive*, *user\_primitive*, *action\_primitive* and *object\_primitive*, each of which defines primitives on one semantic dimension. Properties can exist between different sub-ontologies. For instance, the *hasAuthor* property connects concepts of the *object\_primitive* to those of the *user\_primitive*.

<sup>1</sup> The research is supported by NCET and NSFC

## 2.2 The Archive Expression Model (AEM)

The purpose of AEM model is to formalize the structured information featuring the meeting procedure. We use event, which is an objective description of a significant occurrence or happening located at a certain point in space and time, as the basic organizational entity of the AEM model because most of the interesting information in meetings is related to events. Typical examples of such events include an attendees entering the meeting room or interacting with a meeting artifact. Moreover, it is suggested by the human memory organization theory that events play an important role in organizing data and information. The formalized definition of the AEM model using Extended Backus-Naur Form [18] (EBNF) is shown in figure3. Note that EBNF is used only to illustrate the conceptual structure of AEM model and other formats, e.g., XML, can certainly be leveraged as the binding in real implementation.

```

<Meeting_Procedure> ::= <Event> {, <Event>};
<Event> ::= <Time><Location><User><Action><Object>
<Time> ::= <Begin><End>
<Begin> ::= Non-negative integer
<End> ::= Non-negative integer
<Location> ::= individuals defined in location_primitive
<User> ::= individuals defined in user_primitive
<Action> ::= individuals defined in action_primitive
<Object> ::= individuals defined in object_primitive

```

Figure 3. EBNF definition of AEM model

Similar to the statements in natural language, we adopt a five-dimension structure to describe the semantic of events. The five dimensions are time, location, user, action and object. Time and location describe the temporal and spatial position of the event, while user, action and object contain semantic primitives corresponding to subject, predicate and object in the statement describing an event. For example, the event of Andy’s annotation on document budget.ppt, which can be stated in natural language as “Steve annotates budget.ppt”, contains primitives *Andy*, *annotate* and *budget.ppt* in the user, action and object dimensions respectively.

Although it is still yet to be proved if such a model is the best as a structured representation of meeting procedures, we have observed that similar forms are adopted by many systems that capture meeting or lectures experiences[2][6][10][11]. The AEM model has the following advantages: firstly, it is general. No matter what type the meeting is, it always has a sequential structure in the temporal dimension and can thus be represented by a set of ordered events. Secondly, AEM model is composed of the basic primitives relating to the meeting scenario, thus more implications can be deduced if the primitive ontology is complete. Thirdly, its own structure is simple and can be bound to the XML format, which is both machine-processable and human-readable.

## 2.3 The Query Expression Model (QEM)

Since the structured representation of meeting procedures is a set of sequentially ordered events, the retrieval process can be interpreted as finding the events that match the retrieval queries. Users can get information from the semantic of the selected events

or directly index into other media (video, audio) based on the timestamp to see further details.

Before queries can be processed by machines, they should be formalized so that machines can understand their meaning. Thus the QEM model, which derives from the SQL model, is used in MiF to formalize the retrieval queries to meeting data. A query is similar to the WHERE clause in SQL model which consists of multiple conditions joined by logical operators. Each condition states a criterion to be satisfied. For instance, if the events of *Andy*’s annotating on *budget.ppt* are to be retrieved, the query would be:

*User* equalTo “*Andy*” **AND** *Action* equalTo “*annotate*” **AND** *Object* equalTo “*budget.ppt*” (1)

For each condition, there are three elements: the **subject** denotes the semantic dimension of the event to be concerned; the **value** expresses the primitive to be compared with that contained in subject; the **operator** states a relation between subject and value which is to be satisfied. If such a relation exists between the primitive defined by subject and that defined by value, then the condition is true. An event will be selected into the retrieval result set if the expression of conditions joined by logical operators is true. The QEM model can be formalized in EBNF as illustrated in figure 4.

```

<Query> ::= <Condition> { <Logical_Opr><Condition> }
<Condition> ::= <Subject><Operator><Value>
<Subject> ::= Time | Location | User | Action | Object
<Operator> ::= <Property> | instanceOf
<Property> ::= properties defined in the primitive ontology
<Value> ::= <Class> | <Individual>
<Class> ::= classes defined in the primitive ontology
<Individual> ::= individuals defined in the primitive ontology
<Logical_Opr> ::= AND | OR

```

Figure 4. EBNF definition of QEM model

Since the primitive ontology defines all semantic units involved in a meeting scenario, the value of these three elements should be based on the primitive ontology: **subject** should be one of the five dimensions of the AEM model and will be replaced by explicit primitives when the query is processed; **operator** can be either a property in the primitive ontology or *instanceOf*, the only relation between a class and an individual. The **value** element is more complex for the purpose of extensibility: it can be either a class or an individual of the primitive ontology, or just another query. If the events related to an individual are to be retrieved, an individual will be used as the value element as in (1). Similarly if events are related to a group of individuals featured by the concept of a class, then a class should be used. However, there are situations that relations between multiple events are to be explored, requiring several queries be combined to compose a compound query, and in this case the value element could be another query, which is termed as the inline query. Users may want to use a compound query due to the following reasons:

i) Based on the application scenario, some events, when combined together, may convey information at a higher semantic level. For instance, in a group meeting of our lab if a user opens a document and keeps interacting with document and talking for

more than 10 minutes, this behavior can be confidently interpreted as a presentation. However in a command post, the same behavior may be a report of the current battlefield status.

ii) As reported in [13], some type of memory cues (room layout, attendee names) are better remembered than other type of cues such as meeting date and the exact time of certain events. Thus users sometimes may use events with catchy cues to help them recall other types of events.

iii) Accurate values to be included in a condition are sometimes unknown to the user and should be retrieved by another query. For example, Andy wants to see if he has missed important instructions given by his boss from all the meeting records (Andy often arrives late for the weekly group meeting for a couple of months due to certain reasons). However, he is not sure about the exact time of his entrance so he can first define a query to search for the event of his entering into the meeting room and then includes it as the inline query in another compound query for events featuring the speech given by his boss occurring before his entrance.

Taking the third situation as an example, Andy can first create a query to search for the events of his entrance as in (2) and assign it a unique ID as *AndyEntrance*; then he can make a compound query as expression (3) to retrieve the event he wants (suppose the ID of his boss in the primitive ontology is *Eric*)

User equalTo “Andy” AND Action equalTo “enter” (2)

User equalTo “Eric” AND Action equalTo “talk” AND Time before *AndyEntrance.Time* (3)

Here *AndyEntrance.Time* stands for the *time* attribute of query *AndyEntrance*. The attribute of a query *q* on dimensions *D* is defined as:

$$q.D ::= \{p \mid \exists e, e.D = p \wedge e \text{ satisfy } q\} \quad (4)$$

where *p* is a primitive and *e* is an event. For instance, *AndyEntrance.Time* is a set of the temporal primitives contained in the *time* dimension of the events satisfying the query *AndyEntrance*, i.e., the exact time of Andy’s entrance.

## 2.4 The Query Processor

The query processor will traverse all the events and pick up those matching the retrieval queries by checking if the primitives of an event satisfy the selection criterion. Figure5 is the pseudo code fragment demonstrating this process.

In the case of a compound query, the processor will first process the inline queries and replace those value elements expressed in *query.Dimension* with explicit primitives as equation (4) demonstrates. ( replace the left part with the right part )

After this step, the query evolves to be a set of conditions which are expressed as (D, O, V) and joined by logical operators. (D stands for **dimension**, O stands for **operator** and V for **value expressed in primitives**). For each event *e*, the processor will examine the validity of statement (e.D O V) with the inference engine. e.D is an individual and V can be either a class or an individual of the primitive ontology. If V is a class primitive, it should be inferred if e.D is an instance of V, otherwise the inference would be to examine if (e.D, V) is an instance of property O according to the primitive ontology. Such an inference will be feasible if and only if the primitive ontology retains

computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time).

```

Function name: Query_Process
Input: query // a set of conditions joined by logical operators
Input: record
Output: result

FOR each condition in query
  IF condition.value is the dimension d of another quer
    Set temp_set to the output of Query_Process
    (condition.value, record) // recursive call
  Set condition.value to the temp_set.d
END IF
END FOR
SET result to NULL
FOR each event in the record
  FOR each condition in query
    Set condition.subject to event(condition.subject)
    Set bValid to FALSE
    FOR each primitive in condition.value
      IF reason(condition.subject, condition.operator,
primitive) is TRUE // this step is reasoned by the inference engine
        Set bValid to TRUE
      END IF
    END FOR
    Set condition to bValid
  END FOR
  // now all conditions in query have been replaced with logical
  // values and query has turned to be a boolean expression
  IF query is TRUE
    ADD event to result
  END IF
END FOR

```

Figure 5. The pseudo code for query process

## 3. A meeting record system

### 3.1 Overview

In SEMIC project, we have implemented a meeting capture system (as shown in figure 6) including the functions of live meeting capture, archive, retrieval and playback.

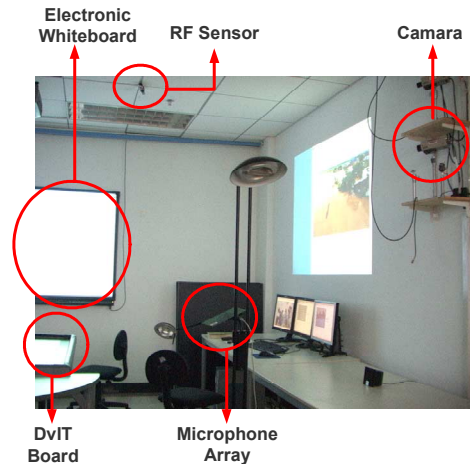


Figure 6. Layout and sensors for meeting capture

Currently the meaningful events that can be detected and archived as structured information include the followings:

- Local speech and the speaker information. (detected by Microphone Array and location-tracking)
- Attendees entering and leaving the meeting room (detected by location-tracking through RFID)
- Dynamic participation of remote user (detected by remote access module)
- Attendee’s interaction with meeting artifacts such as electronic slides and agenda. (detected by the fusion of electronic whiteboard and software technique such as computer vision)

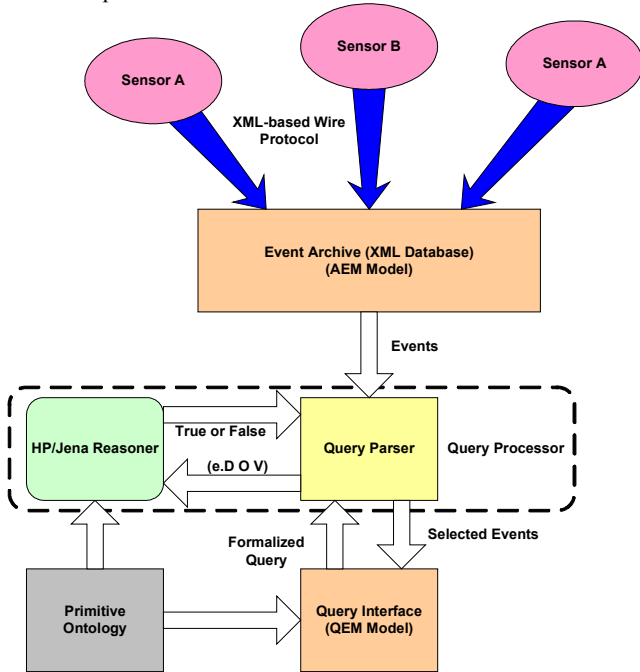


Figure 7. Architecture of the meeting capture and access system and the data flow (SEMIC)

Figure 7 demonstrates the overall architecture of the meeting capture and access system in SEMIC.

### 3.2 The implementation and binding of the components

After the record starts, all events detected by the sensors will be transmitted to a native XML database through XML-based wire protocol and stored in XML documents. XML is leveraged to as the binding of the AEM model for its extensibility and power in expressing semantics that can be understood and processed by machines. Figure 8 is a fragment demonstrating the description of events in XML format.

We have implemented a wizard-based interface for users to create a query based on QEM model. Query creation is divided into several steps and at each step one condition will be defined. As shown in the figure 9, all three elements, subject, operator and value, are selected from combo-boxes and the value element can be either a primitive or another query. Users can join a new condition with the defined conditions by clicking the “AND” or “OR” buttons. Queries can be loaded from or saved into the configure file, thus frequently-used queries can be shared among

```
<Session version="1.0">
  <Context>
    <Duration>2503</Duration>
    <SessionName>GroupMeeting2</SessionName>
  </Context>
  <Procedure>
    <Event EventID="0" User="Shi" Action="talk"
    Object="Schedule" Begin="5" End="17"/>
    <Event EventID="1" User="Chen" Action="openDoc"
    Object="Ontology.ppt" Begin="36" End="38"/>
    <Event EventID="2" User="Qin"
    Action="annotateDoc" Object="Ontology.ppt" Begin="41"
    End="46"/>
    <Event EventID="3" User="He" Action="talk"
    Object="OWL language" Begin="67" End="97"/>
    <Event EventID="4" User="Li" Action="talk"
    Object="N/A" Begin="99" End="107"/>
    <Event EventID="5" User="Jiang" Action="talk"
    Object="OWL language" Begin="111" End="132"/>
  </Procedure>
</Session>
```

Figure 8. Structured description of the meeting procedure based on AEM model

different situations by different users, or used as customized representation of some higher level events. For instance, the query retrieving events of a user’s continuous talking accompanied by frequent interaction with electronic slides in a certain period can be viewed as a representation of giving a presentation—a higher-level event that is not explicitly archived in the record.

The data items of the combo boxes are loaded from the configure files where the primitive ontology is defined. Thus when the meeting scenario changes, e.g., from a brainstorming meeting to a more structured budget report meeting with entirely different semantic primitives (such as attendee ID, action type, agenda, etc.), the only necessary adaptation is to modify the primitive ontology defined in the configure files, making our implementation general and reusable.

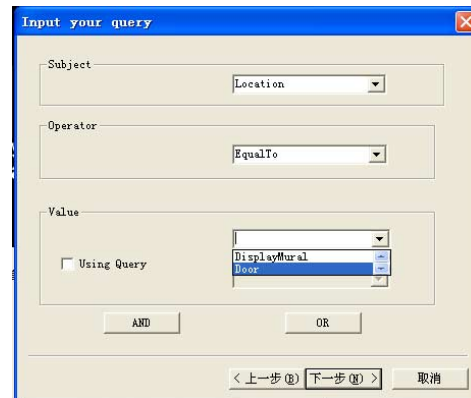


Figure 9. Wizard interface to define an event descriptor

To ensure the computability and decidability, OWL DL sublanguage [16] is used as the binding of the primitive ontology, and HP/Jena [17] reasoner, an open-source tool supporting ontology reasoning, is imported as the inference engine. As shown in figure 10, the defined queries are listed in the left panel and the retrieval result, which is a set of events satisfying the query, is

displayed in the right panel. All events in the result set are visualized in a hierarchical structure under a root node representing the meeting it belongs to, with a combination of its metadata and a key-frame extracted from the video at the temporal position of that event.

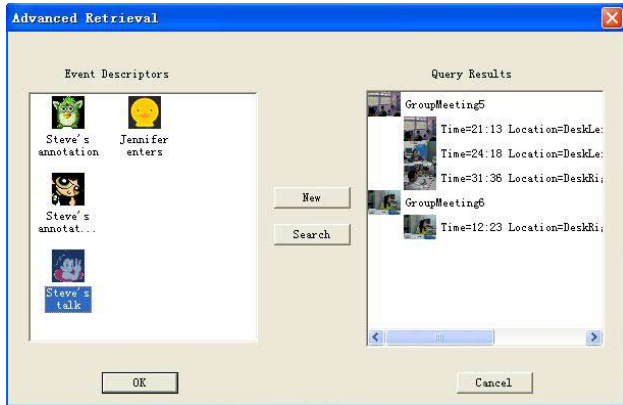


Figure 10. Retrieval interface

### 3.3 A sample of the primitive ontology

The content of the primitive ontology is related to the application semantic and should be reconfigured in different meeting scenarios. We will describe the ontology model we have defined for our weekly group meetings as an example to demonstrate the usage of primitive ontology. For simplicity, we only introduce the *user\_primitive* and its structure can be illustrated in a graphical form as shown in figure 11.

As we can see, the classes and individuals are organized in a hierarchical tree structure with individuals located as leaf nodes. For expression convenience, here we only discuss one property *hasSameInterest*, which states that the two individuals share the same research interest. This property is both symmetric and transitive and the explicitly stated instances of *hasSameInterest* are illustrated in figure 11 as red lines.

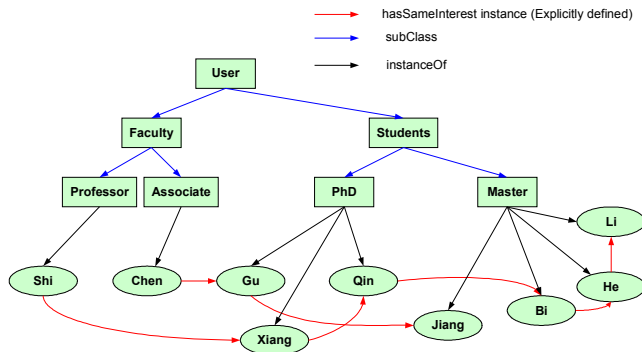


Figure 11. A sample of the primitive ontology

Based on this model, if we want to look for events related to faculties within the record demonstrated in figure 8, the query will return events with ID "0" and "1" since the inference engine can deduce that *Chen* and *Shi* are both faculties from the facts that *Professor* and *Associate* are both subclass of the class *Faculty* and

*Shi* is an instance of *Professor* and *Chen* is an instance of *Associate*.

If *Xiang* has missed this meeting and wants to see what other persons with the same research interests has addressed during the meeting from the record, the query processor will return events with ID "0", "2", "3" and "4". The inference engine will reason that *Shi*, *He* and *Li* all share the same research interests with *Xiang* based on the following facts:

- (1) *Shi hasSameInterest Xiang* because this is explicitly stated;
- (2) *Qin hasSameInterest Xiang* because *Xiang hasSameInterest Qin* and *hasSameInterest* is symmetric;
- (3) *Xiang hasSameInterest Qin* and *Qin hasSameInterest Bi* and *Bi hasSameInterest He* can lead to the conclusion that *Xiang hasSameInterest He* because *hasSameInterest* is transitive. Again, since this property is symmetric, we can get the fact that *He hasSameInterest Xiang*.
- (4) Similar to (3), *Li hasSameInterest Xiang*.

In a more structured and larger meeting the inference functionality would be more crucial because the knowledge size will grow exponentially if everything needs to be explicitly stated. Besides, the inference that can be made are not limited to deductions based on the hierarchical structure or property attribute (transitiveness or symmetry), but can be inferred with multifarious knowledge such as the flexible class definition.

### 4. Conclusion and Future Work

In this paper, we present a semantic framework to formalize the data in meeting procedures and user queries based on the ontology of semantics involved in meetings. With this framework, complex user queries can be expressed and processed for an efficient retrieval and access to the meeting records. A prototype meeting capture and access system has been implemented

Compared with the keyword-based retrieval, the MiF is more expressive and comprehensive. However, we are still not sure how much this expressiveness can benefit users. Thus a future task for us will be to find if the prototype implemented based on MiF will significantly improve the access efficiency or the keyword matching is enough for authentic use. This requires the observation and analysis of the types of query that are frequently made, including their structure, complexity, etc.

### 5. REFERENCES

- [1] Brotherton, J.A., Abowd, G.D. *Lessons Learned From eClass: Assessing Automated Capture and Access in the Classroom*, ACM Trans. Computer-Human Interaction, Vol. 11, No.2, June 2004, Pages 121-155.
- [2] Heather, R.A., Abowd, G.D., Geyer, W., Fuchs, L., Daijavad, S., Poltrock, S. *Integrating Meeting Capture and Access within a Collaborative Team Environment*. In *Proceedings of the 3rd International Conference in Ubiquitous Computing (UbiComp '01)*, 2001, pp. 123-138.
- [3] Gross, R., Bett, M., Yu H., Zhu, X.J., Pan, Y., Yang, J., Waibel, A., *Towards A Multimodal Meeting Record*. In the *Proceedings of IEEE ICME 2000*, 2000.

- [4] Abowd G.D. *Classroom 2000: An experiment with the instrumentation of a living educational environment*. *IBM Systems J.*, Vol. 38(4) pp. 508-530
- [5] Mukhopadhyay, S., Smith, B. *Passive Capture and Structuring of Lectures*. In the *Proceedings of ACM Multimedia '99(MM'99)* 1999, pp. 477-487
- [6] Baldochi L., Andrade A., Cattelan R., Pimentel M. *Architecture and Components for Capture and Access Application*. In the *Proceedings WebMedia and LA-Web*, 2004, pp. 150 – 157
- [7] SEMIC Project Website (Under Construction): <http://media.cs.tsinghua.edu.cn/~pervasive/projects/semic/outline.htm>
- [8] Jain, R., Kim, P., Li, Z. *Experiential Meeting System*. In the *Proceedings of ACM SIGMM 2003 Workshop on Experiential Telepresence (ETP03')*. 2003, pp. 1-12
- [9] Hakeem, A., Shah M. *Ontology and Taxonomy Collaborated Framework for Meeting Classification*. In the *Proceedings of 17<sup>th</sup> international conference on Pattern Recognition. (ICPR'04)* 2004. pp. 1-5
- [10] Hurst W., Muller R., Mayer C. *Multimedia Information Retrieval from Recorded Presentations*. In the *Proceedings of ACM SIGIR'00*. pp. 339-341
- [11] Truong, K., Abowd, G.D. *INCA: A Software Infrastructure to Facilitate the Construction and Evolution of Ubiquitous Capture & Access Applications*. In the *Proceedings of the 2nd International Conference on Pervasive Computing, (Pervasive'04)*.2004, pp. 140-157.
- [12] Hindus, D., Schmandt, C. *Ubiquitous Audio: Capturing Spontaneous Collaboration*. In the *Proceedings of ACM CSCW'92*, 1992, pp. 210-216.
- [13] Jaimes A., Omura K., Nagamine T., Hirata K. *Memory Cues for Meeting Retrieval*. In the *Proceedings of CARPE'04*. ACM. pp. 74-84.
- [14] Nair R. *Calculation of an Aggregated Level of Interest Function for Recorded Events*. In the *Proceedings ACM Multimedia 2004( MM'04)*. ACM.pp.272-275
- [15] Baldochi, L., Cattelan, Pimentel, G., Truong K. *Automatic Generation of Capture and Access Applications*. In the *Proceedings of SBMIDIA 2002: The 8th Brazilian Symposium on Multimedia and Hypermedia Systems*, 2002, pp. 100-115.
- [16] OWL Web Ontology Language  
<http://www.w3.org/TR/2004/REC-owl-features-20040210/#s2.2>
- [17] Jena Semantic Web Toolkit.  
<http://www.hpl.hp.com/semweb/tools.htm>
- [18] Extended Backus-Naur Form.  
<http://www.cee.hw.ac.uk/~rjp/Coursewww/Cwww/EBNF.html>