Fuzzy XML data modeling with the UML and relational data models

Z.M. Ma *, Li Yan

Department of Computer Science and Engineering, Northeastern University, Shenyang, Liaoning 110004, China

Received 21 February 2006; received in revised form 4 November 2006; accepted 7 June 2007
Available online 22 June 2007

Abstract

Information imprecision and uncertainty exist in many real-world applications and for this reason fuzzy data modeling has been extensively investigated in various data models. Currently, huge amounts of electronic data are available on the Internet, and XML has been the de facto standard of information representation and exchange over the Web. This paper focuses on fuzzy XML data modeling, which is mainly involved in the representation model of the fuzzy XML, its conceptual design, and its storage in databases. Based on “possibility distribution theory”, we developed this fuzzy XML data model. We developed this fuzzy UML data model to design the fuzzy XML model conceptually. We investigated the formal conversions from the fuzzy UML model to the fuzzy XML model and the formal mapping from the fuzzy XML model to the fuzzy relational databases. © 2007 Elsevier B.V. All rights reserved.

Keywords: Fuzzy sets and possibility distributions; XML; UML; Relational databases; Conceptual design; Mapping

1. Introduction

With the wide utilization of the Web and the availability of huge amounts of electronic data, information representation and exchange over the Web becomes important, and XML has been the de facto standard [7]. XML and related standards are technologies that allow the easy development of applications that exchange data over the Web such as e-commerce (EC) and supply chain management (SCM). This creates a new set of data management requirements involving XML, such as the need to store and query XML documents. To store, query and update XML data, it is necessary to integrate XML and databases [3]. Various databases, including relational, object-oriented, and object-relational databases, have been used for mapping to and from the XML document [10,16–19,38,45]. Among these kinds of databases, relational databases might be the more...
promising alternative because of the widespread use and mature techniques [13]. Moreover, XML lacks sufficient power in modeling real-world data and their complex inter-relationships in semantics. Thus, it is necessary to use other methods to describe data paradigms and develop a true conceptual data model, and then transform this model into an XML encoded format, which can be treated as a logical model. Conceptual data modeling of XML document schema [11,35,27,48,14] and XML schema [2,37] have been studied in the recent past. In [11], for example, UML was used for designing XML DTD (document type definition). The idea is to use essential parts of the static UML to model the XML DTD. The mapping between the static part of the UML (i.e., class diagrams) and the XML DTDs was developed. To take advantage of all facets that DTD concepts offer, the authors further extended the UML language in an UML-compliant way.

In real-world applications, however, information is often vague or ambiguous. Therefore, different kinds of imperfect information [44] have been extensively introduced into databases. Inconsistency, imprecision, vagueness, uncertainty, and ambiguity are five basic kinds of imperfect information in database systems [6].

- Inconsistency is a kind of semantic conflict, meaning the same aspect of the real world is irreconcilably represented more than once in a database or in several different databases. For example, the age of George is stored as 34 and 37 simultaneously. Information inconsistency usually comes from information integration [12].
- Of course, imprecision and vagueness are relevant to the content of an attribute value, meaning that a choice must be made from a given range (interval or set) of values without knowing which one to choose. In general, vague information is represented by linguistic values. For example, the age of Michael is a set $\{18,19,20,21\}$, a piece of imprecise information, and the age of John is a linguistic “old”, being a piece of vague information.
- The uncertainty is related to the degree of truth of its attribute value, meaning that we can apportion some, but not all, of our belief to a given value or group of values. For example, the possibility that the age of Chris is 35 right now should be 98%. The random uncertainty, described using probability theory, is not considered in the paper.
- The ambiguity means that some elements of the model lack complete semantics, leading to several possible interpretations.

Generally, several different kinds of imperfection can co-exist with respect to the same piece of information. For example, the age of Michael is a set $\{18,19,20,21\}$ and their possibilities are $70\%$, $95\%$, $98\%$, and $85\%$, respectively. Also Smets [44] presents some aspects of imperfection, in which imprecision, inconsistency and uncertainty are the major groups. Imprecision and inconsistency are essential properties of the information itself, whereas uncertainty is a property of the relation between the information and our knowledge about the world. To model imprecision and uncertainty, the various approaches are presented in [44]. These models are grouped into two large categories, namely, the symbolic and the quantitative models. Fuzzy sets introduced by Zadeh [52] have been widely used for the quantification of imprecision and uncertainty.

Fuzzy information has been extensively investigated in the context of the relational model [8,34,36]. In order to model uncertain data and complex-valued attributes as well as complex relationships among objects, current efforts are focusing on the conceptual data models [51,26], and object-oriented databases [22], with imprecise and uncertain information. Information fuzziness has also been investigated in the context of e-commerce (EC) and supply chain management (SCM) [33,50,49]. It is shown that fuzzy set theory is very useful in Web-based business intelligence. Unfortunately, although it is the current standard for data representation and exchange over the Web, XML is not able to represent and process imprecise and uncertain data, although the databases with imprecise and uncertain information have been extensively discussed. Currently, little research has been done in modeling and querying imperfect XML data. Only XML with incomplete information [1] and probabilistic data in XML [30,47] has been proposed in research papers. More recently, Lee and Fanjiang [20] developed a fuzzy OO modeling technique schema based on XML to model requirement specifications and incorporate the notion of stereotype to facilitate the modeling of imprecise requirements.

In this paper, we identify multiple granularity of data fuzziness in XML. Based on possibility distribution theory, we developed a fuzzy XML data model that addresses all types of fuzziness. Also, we developed a fuzzy UML data model to design the fuzzy XML data model conceptually. In particular, we investigated
the formal conversions from the fuzzy UML model to the fuzzy XML model, and the formal mapping from the fuzzy XML model to the fuzzy relational databases. Regarding the practical issues of fuzzy UML, the representation of fuzziness in UML associations was dealt with in [42], and a practical approach and implementation for a fuzzy-UML storage version were described in [43]. However, it should be noted that the fuzzy UML in this paper does not aim to provide an “official” extension to the UML.

The remainder of this paper is organized as follows: Section 2 gives basic knowledge concerning fuzzy relational databases based on fuzzy sets and possibility distribution theories. Section 3 investigates data fuzziness in an XML document, and a fuzzy representation model for fuzzy data in XML is thereby proposed. The fuzzy extension to the UML class model and conceptual design of the fuzzy XML with the fuzzy UML are developed in Section 4. The mapping from the fuzzy XML model to the fuzzy relational databases is presented in Section 5. Section 6 concludes this paper.

2. Fuzzy sets and the fuzzy relational databases

The fuzzy set was originally introduced by Zadeh in 1965 [52]. Let $U$ be a universe of discourse and $F$ be a fuzzy set in $U$. A membership function $\mu_F: U \rightarrow [0,1]$ is defined for $F$, where $\mu_F(u)$, for each $u \in U$, denotes the membership degree of $u$ in the fuzzy set $F$. Thus, the fuzzy set $F$ is described as follows:

$$F = \{\mu_F(u_1)/u_1, \mu_F(u_2)/u_2, \ldots, \mu_F(u_n)/u_n\}$$

when $U$ is not a discrete set, the fuzzy set $F$ can be represented by

$$F = \int_{u \in U} \mu_F(u)/u.$$ 

Note that, in this paper, $\mu_F$ is used to represent the membership function of fuzzy set $F$, and $\mu_F(u)$ is used to represent the membership degree that $u$ belongs to fuzzy set $F$. This paper only concentrates on discrete fuzzy sets.

When the $\mu_F(u)$ above is explained to be a measure of the possibility that a variable $X$ has the value $u$ in this approach, where $X$ takes values in $U$, a fuzzy value is described by a possibility distribution $\pi_X$ [53].

$$\pi_X = \{\pi_X(u_1)/u_1, \pi_X(u_2)/u_2, \ldots, \pi_X(u_n)/u_n\}$$

Here, $\pi_X(u_i), u_i \in U$ denotes the possibility that $u_i$ is true. A fuzzy set is a representation of a concept while possibility distribution relates with the possibility of occurring a value within a distribution. Let $\pi_X$ and $F$ be the possibility distribution representation and the fuzzy set representation for a fuzzy value, respectively. It is clear that $\pi_X = F$ is true [36].

By means of fuzzy sets and possibility distributions, a fuzzy value on $U$ can be characterized by a fuzzy set or a possibility distribution in $U$. Also, information fuzziness can be described by means of similarity relations in domain elements [8], in which the fuzziness comes from the similarity relations between individual values in a universe of discourse, not from the status of an object itself. Similarity relations are used to describe the similarity degree of individual values from the same universe of discourse. Thus, a fuzzy value is hereby represented by a set in which the elements are some individual values of the universe of discourse and there is a similarity relation in the universe of discourse. Let us look at an example in [51]. Assume that we have a universe of discourse “Popularity” with domain \{very-popular, popular, mod-popular, not-popular\} and a similarity relation shown in Table 1. Then a fuzzy value “more or less popular” may be represented by \{very-popular, popular, mod-popular\}.

<table>
<thead>
<tr>
<th></th>
<th>very-popular</th>
<th>popular</th>
<th>mod-popular</th>
<th>not-popular</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very-popular</td>
<td>1.0</td>
<td>0.8</td>
<td>0.6</td>
<td>0</td>
</tr>
<tr>
<td>Popular</td>
<td>0.8</td>
<td>1.0</td>
<td>0.8</td>
<td>0.1</td>
</tr>
<tr>
<td>Mod-popular</td>
<td>0.6</td>
<td>0.8</td>
<td>1.0</td>
<td>0.4</td>
</tr>
<tr>
<td>Not-popular</td>
<td>0</td>
<td>0.1</td>
<td>0.4</td>
<td>1.0</td>
</tr>
</tbody>
</table>
A similarity relation $Sim$ on the universe of discourse $U$ is a mapping: $U \times U \rightarrow [0,1]$ so that

(a) for $\forall x \in U$, $Sim(x,x) = 1$ (reflexivity)

(b) for $\forall x, y \in U$, $Sim(x,y) = Sim(y,x)$, and (symmetry)

(c) for $\forall x, y, z \in U$, $Sim(x,z) \geq \max_x (\min(Sim(x,y), Sim(y,z)))$ (transitivity)

In addition to the similarity relation, other relations such as proximity relation [41], resemblance relation [39], and closeness relation [9] are also applied to describe information fuzziness. Compared with the similarity relation, the resemblance relation, for example, only holds the properties of reflexivity and symmetry.

Fuzzy set and possibility distribution theories have been used to extend various database models and this has resulted in numerous contributions, mainly with respect to the popular relational model or to some related form of it. A relation instance $r$ on a relational schema $R(A_1, A_2, \ldots, A_n)$ is a subset of the Cartesian product of $\text{Dom}(A_1) \times \text{Dom}(A_2) \times \cdots \times \text{Dom}(A_n)$, in which $\text{Dom}(A_i)$ is the domain of attribute $A_i$. Therefore, a relation instance can be viewed as a table whose rows and columns are called the tuples and attributes of the relation, respectively. In connection to the three types of fuzzy data representations, which are the fuzzy set representation, the possibility distribution representation, and the similarity relation representation, several approaches have been taken to incorporate fuzzy data into relational databases. One of the fuzzy relational database models is based on fuzzy relation [36], and similarity relation [8] (or proximity [41] and resemblance [39] relations). The other one is based on possibility distribution [34], which can further be classified into two categories: tuples associated with possibilities and attribute values represented by possibility distributions.

Now let us look at the tuples of fuzzy relational databases in order to understand the fuzzy relational database models. The form of an $n$-tuple in each of the above-mentioned fuzzy relational models can be expressed, respectively, as follows:

For type 1 fuzzy relational model:

$$t = \langle p_1, p_2, \ldots, p_i, \ldots, p_n \rangle$$

where $p_i \subseteq \text{Dom}(A_i)$ with $\text{Dom}(A_i)$ being the domain of attribute $A_i$. For each $\text{Dom}(A_i)$, there exists a similarity (or proximity and resemblance) relation.

For type 2 fuzzy relational model:

$$t = \langle a_1, a_2, \ldots, a_i, \ldots, a_n, d \rangle$$

and

For type 3 fuzzy relational model:

$$t = \langle \pi_{A_1}, \pi_{A_2}, \ldots, \pi_{A_i}, \ldots, \pi_{A_n} \rangle$$

Here $a_i \in \text{Dom}(A_i)$ and $d \in (0,1]$. $\pi_{A_i}$ is the possibility distribution of attribute $A_i$ on its domain $\text{Dom}(A_i)$, and $\pi_{A_i}(x)$, $x \in \text{Dom}(A_i)$, denotes the possibility that $x$ is the actual value of $A_i$.

Consider the three fuzzy relations $r_1$, $r_2$ and $r_3$ in Fig. 1. In $r_1$, the values of attribute $\text{Age}$ of the tuples are fuzzy values represented by the similarity relation given in Table 1. Relation $r_1$ is based on the fuzzy relational model of type 1. In $r_2$, each tuple is associated with a possibility degree but all attribute values of the tuples are crisp. Relation $r_2$ is based on the fuzzy relational model of type 2. In $r_3$, no tuples are associated with possibility degree but the values of attribute $\text{Age}$ of the tuples are all fuzzy. Relation $r_3$ is based on the fuzzy relational model of type 3.

Based on the above-mentioned basic fuzzy relational database models, there are several extended fuzzy relational database models. It is clear that one can combine two kinds of fuzziness in possibility-based fuzzy relational databases where attribute values may be possibility distributions, and tuples are connected with membership degrees. Such fuzzy relational databases are called possibility-distribution-fuzzy relational models in [46]. Another possible extension is to combine possibility distribution and similarity (proximity or resemblance) relation, and the extended possibility-based fuzzy relational databases proposed in [9,24], where possibility distribution and resemblance relation arise in a relational database simultaneously.
This paper focuses on the fuzzy relational databases where $n$-tuples have the following form:

$$t = (\pi_{A_1}, \pi_{A_2}, \ldots, \pi_{A_i}, \ldots, \pi_{A_n}, d).$$

That is, each tuple may be associated with a possibility degree, and its attribute values may be fuzzy ones represented by possibility distributions. The formal definition of this kind of fuzzy relational database is given as follows:

**Definition.** A fuzzy relation $r$ on a relational schema $R (A_1, A_2, \ldots, A_n, A_{n+1})$ is a subset of the Cartesian product of $\text{Dom}(A_1) \times \text{Dom}(A_2) \times \cdots \times \text{Dom}(A_n) \times \text{Dom}(A_{n+1})$, where $\text{Dom}(A_i)(1 \leq i \leq n)$ may be a fuzzy subset or even a set of fuzzy subset and $\text{Dom}(A_{n+1})$ is $(0,1]$.

It should be pointed out, however, that although several fuzzy relational database models are proposed in the literature according to the fuzzy data representations, the fuzziness identified in the fuzzy relational databases only have two kinds, namely, the fuzziness in attribute values and the fuzziness in tuples.

3. The fuzzy XML model

In this section, the fuzziness in the XML document is investigated and the representation model of the fuzzy XML document is developed. Here, we focus on XML DTD and develop the extended XML DTD for the fuzzy XML document.

3.1. The fuzziness in the XML document

Two kinds of fuzziness can be found in a relational model: one is to associate membership degrees with individual tuples, and the other is to represent attribute values with possibility distributions. A membership degree associated with a tuple is interpreted to mean the possibility of the tuple being a member of the corresponding relation. A possibility distribution represented as an attribute value means we do not know a crisp value of the attribute but only know the range of values that the attribute may take and the possibility of each value being true.

XML data are structured, and XML can represent imprecise and uncertain information naturally. In the case of XML, we may have membership degrees associated with elements. It is also possible to associate possibility distributions with attribute values of elements. XML restricts attributes to only a unique single value. We modify the schema in XML to make any attribute into a sub-element.

Now let us interpret what a membership degree associated with an element means, given that the element can nest under other elements, and more than one of these elements may have an associated membership degree. The existential membership degree associated with an element should be the possibility that the state of the world includes this element and the sub-tree rooted at it. For an element with the sub-tree rooted at it, each node in the sub-tree is not treated as independent but dependent upon its root-to-node chain. Each possibility in the source XML document is assigned conditioned on the fact that the parent element exists certainly. In other words, this possibility is a relative one based upon the assumption that the possibility the parent element exists is exactly 1.0. In order to calculate the absolute possibility, we must consider the relative possibility in the parent element. In general, the absolute possibility of an element $e$ can be obtained by multiplying the relative possibilities found in the source XML along the path from $e$ to the root. Of course, each of these relative possibilities will be available in the source XML document. By default, relative possibilities are therefore regarded as 1.0.

Consider a chain $A \rightarrow B \rightarrow C$ from the root node $A$. Assume that the source XML document contains the relative possibilities $\text{Poss}(C|B)$, $\text{Poss}(B|A)$, and $\text{Poss}(A)$, associated with the nodes $C$, $B$, and $A$, respectively. Then we have

$$\text{Poss}(B) = \text{Poss}(B|A) \times \text{Poss}(A) \quad \text{and} \quad \text{Poss}(C) = \text{Poss}(C|B) \times \text{Poss}(B|A) \times \text{Poss}(A).$$

Here, $\text{Poss}(C|B)$, $\text{Poss}(B|A)$, and $\text{Poss}(A)$ can be obtained from the source XML document.

For attribute values of elements, XML restricts attributes to a unique single value. It is not difficult to find that this restriction does not always hold true. It is often the case that some data item is known to have multiple values—these values may be unknown completely and can be specified with a possibility distribution. For
example, the e-mail address of a person may be multiple character strings because he or she has several e-mail addresses available simultaneously. In case we do not have complete knowledge of the e-mail address for “Tom Smith”, we may say that the e-mail address may be “TSmith@yahoo.com” with possibility 0.60, “Tom_Smith@yahoo.com” with possibility 0.85, “Tom_Smith@hotmail.com” with possibility 0.85, “TSmith@hotmail.com” with possibility 0.55, and “TSmith@msn.com” with possibility 0.45. In contrast, some data items are known to have a single unique value. For instance, the age of a person in years is a unique non-negative integer. If such a value is unknown so far, we can use the following possibility distribution: \{0.4/23, 0.6/25, 0.8/27, 1.0/29, 1.0/30, 1.0/31, 0.8/33, 0.6/35, 0.4/37\}. Based on the discussion above, it is clear that we have two interpretations for a fuzzy data represented by a possibility distribution: fuzzy disjunctive data and fuzzy conjunctive data.

In summary, we have two kinds of fuzziness in an XML document:

• the first is the fuzziness in elements, and we use membership degrees associated with such elements;
• the second is the fuzziness in attribute values of elements, and we use possibility distribution to represent such values.

Note that, for the latter, there exist two types of possibility distribution, i.e., disjunctive and conjunctive possibility distributions, and they may occur in child elements with or without further child elements in the ancestor–descendant chain.

Fig. 2 gives a fragment of an XML document with fuzzy information.

In the example above, we talk about the universities in an area of a given city, say, Detroit, Michigan, in the USA. Wayne State University is located in downtown Detroit, and the possibility that it is included in the universities in Detroit is 1. Oakland University, however, is located in a nearby county of Michigan, named Oakland. Whether Oakland University is included in the universities in Detroit depends on how to define the area of Detroit, the Greater Detroit Area or only The City of Detroit. Assume that it is unknown and the possibility that Oakland University is included in the universities in Detroit is assigned 0.8. In addition, an employee, Frank Yager, at Oakland University is under the stage of promotion and his position may be associate professor or professor. The possibility that he is an associate professor, teaches a course called Advances in Database Systems, and occupies the office called B1024 is 0.8. The possibility that he is a professor, teaches a course called Advances in Database Systems, and occupies the office called B1024 is 0.6. As to the student Tom Smith, he has fuzzy values in the attributes age and email, which are represented by a disjunctive possibility distribution and conjunctive possibility distribution, respectively.

3.2. The representation model

3.2.1. Representation of fuzzy data in the XML document

It is apparent from the example given above that a possibility attribute, denoted Poss, which takes a value of \([0,1]\), should be introduced first. This possibility attribute is applied together with a fuzzy construct called Val to specify the possibility of a given element existing in the XML document.

Consider Line 3 of Fig. 2, \(<Val \text{ Poss} = 0.8>\), where it is stated that the possibility of the given element university being Oakland University is equal to 0.8. For an element with possibility 1.0, pair \(<Val \text{ Poss} = 1.0>\) and \(</Val>\) is omitted from the XML document.

Based on pair \(<Val \text{ Poss}>\) and \(</Val>\), possibility distribution for an element can be expressed. Also, possibility distribution can be used to express fuzzy element values. For this purpose, we introduce another fuzzy construct called Dist to specify a possibility distribution. Typically, a Dist element has multiple Val elements as children, each with an associated possibility. Since we have two types of possibility distribution, the Dist construct should indicate the type of a possibility distribution being disjunctive or conjunctive.

Again consider Fig. 2. Lines 24–34 are the disjunctive Dist construct for the age of student “Tom Smith”. Lines 38–44 are the conjunctive Dist construct for the email of student “Tom Smith”. It should be noted, however, that the possibility distributions in Lines 24–34 and 38–44 are all for leaf nodes in the ancestor–descendant chain. In fact, we can also have possibility distributions and values over non-leaf nodes. Observe the disjunctive Dist construct in Lines 6–19, which express the two possible statuses for the employee with ID...
In these two employee values, Lines 7–12 are with possibility 0.8, and Lines 13–18 with possibility 0.6.

3.2.2. DTD modification

It has been shown that the XML document must be extended for fuzzy data modeling. As a result, several fuzzy constructs have been introduced. In order to accommodate these fuzzy constructs, it is clear that the DTD of the source XML document should be correspondingly modified. In this section, we focus on DTD modification for fuzzy XML data modeling.
First we define Val element as follows:

```xml
<!ELEMENT Val (#PCDATA) original-definition>
<!ATTLIST Val Poss CDATA "1.0">
```

Then we define Dist element as follows:

```xml
<!ELEMENT Dist (Val+)>
<!ATTLIST Dist type (disjunctive|conjunctive) "disjunctive">
```

Now we modify the element definition in the classical DTD so that all of the elements can use possibility distributions (Dist). For a leaf element which only contains text or #PCDATA, say, leafElement, its definition in the DTD is changed from

```xml
<!ELEMENT leafElement (#PCDATA)>
```

to

```xml
<!ELEMENT leafElement (#PCDATA)Dis>.
```

That is, leaf element leafElement may be a crisp one (e.g., sname of student in Fig. 2), and then could be defined as

```xml
<!ELEMENT leafElement (#PCDATA)>. 
```

Also, it is possible that leaf element leafElement may be a fuzzy one, taking a value represented by a possibility distribution (e.g., age of student in Fig. 2). Then it may be defined as

```xml
<!ELEMENT leafElement (Dist)>.
```

Furthermore, we have the following definition.

```xml
<!ELEMENT Dist (Val+)>
<!ATTLIST Dist type (disjunctive|conjunctive) "disjunctive">
<!ELEMENT Val (#PCDATA)>
<!ATTLIST Val Poss CDATA "1.0">
```

For the non-leaf element, say nonleafElement, first we should change the element definition from

```xml
<!ELEMENT nonleafElement (original-definition)>to
<!ELEMENT nonleafElement (original-definition)Val+] Dist>,
```

and then add

```xml
<!ELEMENT Val (original-definition)>.
```

That is, the non-leaf element nonleafElement may be crisp (e.g., student in Fig. 2) and then may be defined as

```xml
<!ELEMENT nonleafElement (original-definition)>.
```

When the non-leaf element nonleafElement is a fuzzy one, we differentiate two situations: the element takes a value connected with a possibility degree (e.g., university in Fig. 2), and, second, the element takes a set of values and each value is connected with a possibility degree (e.g., employee in Fig. 2). The former element is defined as follows:
The latter element is defined as

```xml
<!ELEMENT nonleafElement (Dist)>  
<!ELEMENT Dist (Val+)>
  <!ATTLIST Dist type (disjunctive conjunctive) "disjunctive">  
<!ELEMENT Val (original-definition)>  
<!ATTLIST Val Poss CDATA "1.0"/>
```

Then the DTD of the XML document in Fig. 2 is shown in Fig. 3.

4. Conceptual design of the fuzzy XML DTD with the fuzzy UML data model

This section presents the issues on how to conceptually design the fuzzy XML DTD with the UML class diagram. To this purpose, we should first extend the classical UML data model for fuzzy data modeling.

4.1. The fuzzy UML data model

UML provides a collection of models to capture the many aspects of a software system [5,29,31]. Note that while the UML reflects some of the best OO modeling experiences available, it suffers from deficiency of some necessary semantics. One of the deficiencies can be generalized as the need to handle imprecise and uncertain information although such information exists in knowledge engineering and databases and has extensively been studied [32]. In the following, we propose the fuzzy UML data model.

```xml
<!ELEMENT universities (university*)>
<!ELEMENT university (Val+)>
  <!ATTLIST university UName IDREF #REQUIRED>
<!ELEMENT Val (department*)>
  <!ATTLIST Val Poss CDATA "1.0">  
<!ELEMENT department (employee*, student*)>
  <!ATTLIST department DName IDREF #REQUIRED>
<!ELEMENT employee (Dist)>  
  <!ATTLIST employee FID IDREF #REQUIRED>
<!ELEMENT Val (fname?, position?, office?, course?)>
  <!ATTLIST Val Poss CDATA "1.0">  
<!ELEMENT student (sname?, age?, sex?, email?)>
  <!ATTLIST student SID IDREF #REQUIRED>
<!ELEMENT fname (#PCDATA)>  
<!ELEMENT position (#PCDATA)>  
<!ELEMENT office (#PCDATA)>  
<!ELEMENT course (#PCDATA)>  
<!ELEMENT sname (#PCDATA)>  
<!ELEMENT age (Dist)>  
  <!ATTLIST Dist type (disjunctive)>  
<!ELEMENT sex (#PCDATA)>  
<!ELEMENT email (Dist)>  
<!ELEMENT Dist (Val+)>
  <!ATTLIST Dist type (conjunctive)>  
<!ELEMENT Val (#PCDATA)>  
<!ATTLIST Val Poss CDATA "1.0"/>
```

Fig. 3. The DTD of the fuzzy XML document in Fig. 2.
4.1.1. Fuzzy class

The objects having the same properties are gathered into classes that are organized into hierarchies. Theoretically, a class can be considered from two different viewpoints:

(a) an extensional class, where the class is defined by the list of its object instances ("objects" for short), and
(b) an intensional class, where the class is defined by a set of attributes and their admissible values.

Objects model real-world entities or abstract concepts. Objects have properties that may be attributes of the object itself or relationships, also known as associations, between the object and one or more other objects. An object is fuzzy because of a lack of information. For example, an object representing a part in preliminary design will definitely be made of either stainless steel, molded steel, or alloy steel (each of them may be connected with a possibility, say, of 0.7, 0.5 and 0.9, respectively). Formally, objects that have at least one attribute whose value is a fuzzy set are fuzzy objects. A class is fuzzy because of the following several reasons:

(a) A class is extensionally defined, where some objects with similar properties are fuzzy ones. Then the objects belong to the class with membership degree of \([0,1]\).
(b) When a class is intensionally defined, the domains of some attributes may be fuzzy, and thus a fuzzy class is formed.
(c) The subclass produced by a fuzzy class by means of specialization, and the super-class produced by some classes (in which there is at least one class who is fuzzy) by means of generalization, are also fuzzy.

Following on the footsteps of Zvieli and Chen [54], we introduce three levels of fuzziness to the UML classes. These three levels of fuzziness are defined as follows:

(a) At the first level, classes and attribute sets of class may be fuzzy, i.e., they have a possibility to the model.
(b) The second level is related to the fuzzy occurrences of objects.
(c) The third level concerns the fuzzy values of attributes of special objects.

In order to model the first level of fuzziness, i.e., an attribute or a class with possibility, the attribute or class name should be followed by a pair of words WITH mem DEGREE, where \(0 \leq \text{mem} \leq 1\) is a scalar and used to indicate the degree that the attribute belongs to the class or the class belongs to the data model [28]. Assume, for example, that we have a class "Employee" with attribute "Cell Phone" in a data model about students in universities. This data model contains some classes, and the class "Employee" may or may not need to be included in the data model. In addition, the class "Employee" consists of some attributes, and the attribute "Cell Phone" may or may not need to be included in the class. Assume we have "Employee WITH 0.8 DEGREE" and "Cell Phone WITH 0.6 DEGREE", which mean the class and attribute with the first level of fuzziness, respectively. Here, class "Employee" belongs to the data model with 0.8 degree while attribute "Cell Phone" belongs to the class with 0.6 degree. Generally, an attribute or a class will not be declared when its degree is 0. In addition, "WITH 1.0 DEGREE" can be omitted when the degree of an attribute or a class is 1. Also note that attribute values may be fuzzy. In order to model the third level of fuzziness, a keyword FUZZY is introduced and is placed in the front of the attribute. As to the second level of fuzziness, we must indicate the degree of possibility that an object of the class belongs to the class. To this purpose, an additional attribute is introduced into the class to represent object membership degree to the class, which attribute domain is \([0,1]\). We denote such a special attribute with \(\mu\) in this paper. In order to differentiate the class with the second level of fuzziness, we use a dashed-outline rectangle to denote such a class.

Fig. 4 shows a fuzzy class Student. Here, attribute Age may take fuzzy values, namely, its domain is fuzzy. Generally, students may or may not have their own offices. So when class Student is designed, it is unknown for sure if attribute Office should be included in class Student. In other words, attribute Office uncertainly belongs to the class Students. A possibility, say 0.4, is assigned to attribute Office with regard to class Student. We use "with 0.4 possibility degree" to describe the fuzziness at the first level in the class definition. In addition, we may not determine if an object is the instance of the class because the class is fuzzy. So an additional attribute \(\mu\) is introduced into the class.
4.1.2. Fuzzy generalization

The concept of subclassing is one of the basic building blocks of the object model. A new class, called subclass, is produced from another class, called superclass by means of inheriting all attributes and methods of the superclass, overriding some attributes and methods of the superclass, and defining some new attributes and methods. Since a subclass is the specialization of the superclass, any object belonging to the subclass must belong to the superclass. This characteristic can be used to determine if two classes have a subclass/superclass relationship.

However, a class produced from a fuzzy class must be fuzzy. If the former is still called subclass and the latter superclass, the subclass/superclass relationship is fuzzy. In other words, a class is a subclass of another class with membership degree of $[0,1]$ at this moment. We have the following criteria to determine the fuzzy subclass/superclass relationship.

(a) For any (fuzzy) object, say $e$, let the membership degree that it belongs to the subclass, say $B$, be $\mu_B(e)$ and the membership degree that it belongs to the superclass, say $A$, be $\mu_A(e)$. Then $\mu_B(e) \leq \mu_A(e)$.

(b) Assume that a threshold, say $\beta$, is given. Then $\mu_B(e) \geq \beta$. Here $B$, $e$, and $\mu_B(e)$ are the same as the above.

The subclass $B$ is then a subclass of the superclass $A$ with a membership degree. This membership degree is the minimum of the membership degrees to which these objects belong to the subclass. Here the given threshold is used for a computational threshold to avoid propagating infinitesimal degrees (the same in the rest of this paper). Formally, we have the following definition for the fuzzy subclass/superclass relationship.

**Definition.** Let $A$ and $B$ be two (fuzzy) classes with object membership functions $\mu_A$ and $\mu_B$, respectively. Let $\beta$ be a given threshold. We say $B$ is a subclass of $A$ if

\[
(\forall e) \ (\beta \leq \mu_B(e) \leq \mu_A(e)).
\]

The membership degree that $B$ is a subclass of $A$ should be $\min_{\mu_B(e) \geq \beta(\mu_B(e))}$. Here, $e$ is object instance of $A$ and $B$ in the universe of discourse, and $\mu_A(e)$ and $\mu_B(e)$ are membership degrees of $e$ to $A$ and $B$, respectively.

It should be noted, however, that in the above-mentioned fuzzy generalization relationship, we assume that classes $A$ and $B$ can only have the second level of fuzziness. It is possible that classes $A$ or $B$ are the classes with membership degree denoted by scalar, namely, with the first level of fuzziness.

**Definition.** Let two classes $A$ and $B$ be $A$ WITH $degree_A$ DEGREE and $B$ WITH $degree_B$ DEGREE. Here, $degree_A$ and $degree_B$ are the scalars of membership degree. Let $\mu_A$ and $\mu_B$ be the object membership functions of $A$ and $B$, respectively. Then $B$ is a subclass of $A$ if

\[
(\forall e) \ (\beta \leq \mu_B(e) \leq \mu_A(e)) \land (\beta \leq degree_B \leq degree_A).
\]

That means that $B$ is a subclass of $A$ only if, in addition to the condition that the membership degrees of all objects to $A$ and $B$ must be greater than or equal to the given threshold, and the membership degree of any object to $A$ must be greater than or equal to the membership degree of this object to $B$, the membership degrees of $A$ and $B$ must be greater than or equal to the given threshold, and the membership degree of $A$ must be greater than or equal to the membership degree of $B$. 

\[\text{Fig. 4. A fuzzy class in the fuzzy UML.}\]
Consider a fuzzy superclass $A$ and its fuzzy subclasses $B_1, B_2, \ldots, B_n$ with object membership functions $\mu_A, \mu_{B_1}, \mu_{B_2}, \ldots, \mu_{B_n}$, respectively, which may also have the scalars of membership degree $\text{degree}_A, \text{degree}_{B_1}, \text{degree}_{B_2}, \ldots, \text{degree}_{B_n}$, respectively. Then the following relationship is true:

$$\forall e \left( \max(\mu_{B_1}(e), \mu_{B_2}(e), \ldots, \mu_{B_n}(e)) \leq \mu_A(e) \right) \land \left( \max(\text{degree}_{B_1}, \text{degree}_{B_2}, \ldots, \text{degree}_{B_n}) \leq \text{degree}_A \right).$$

It can be seen that we can assess fuzzy subclass/superclass relationships by utilizing the inclusion degree of objects to the class. Clearly such an assessment is based on the extensional viewpoint of class. When classes are defined with the intensional viewpoint, there is no object available. Therefore, the method given above cannot be used. At this point, we can use the inclusion degree of a class with respect to another class to determine the relationships between fuzzy subclass and superclass. The basic idea is that, since any object belonging to the subclass should belong to the superclass, the common attribute domains of the superclass should include the common attribute domains of the subclass.

**Definition.** Let $A$ and $B$ be (fuzzy) classes and the degree that $B$ is the subclass of $A$ be denoted by $\mu(A, B)$. For a given threshold $\beta$, we say $B$ is a subclass of $A$ if

$$\mu(A, B) \geq \beta.$$

Here, $\mu(A, B)$ is a scalar and used to calculate the inclusion degree of $B$ with respect to $A$ according to the inclusion degree of the attribute domains of $B$ with respect to the attribute domains of $A$ as well as the weight of attributes.

To figure out or estimate the inclusion degree of two classes, one needs to know the (fuzzy) attribute domains of the two classes and the weight of the attributes. The problem of evaluating the inclusion degree is outside the scope of the current paper. One can refer to Ma et al. [25], where the methods for evaluating the inclusion degree of fuzzy attribute domains and further evaluating the inclusion degree of a subclass with respect to the superclass are discussed in detail.

Now let us consider the situation that classes $A$ or $B$ are the classes with scalars of membership degree, namely, with the first level of fuzziness.

**Definition.** Let two classes $A$ and $B$ be $A$ WITH $\text{degree}_A$ DEGREE and $B$ WITH $\text{degree}_B$ DEGREE. Here, $\text{degree}_A$ and $\text{degree}_B$ are the scalars of membership degree. Then $B$ is a subclass of $A$ if

$$(\mu(A, B) \geq \beta) \land (\beta \leq \text{degree}_B \leq \text{degree}_A).$$

That means that $B$ is a subclass of $A$ only if, in addition to the condition that the inclusion degree of $A$ with respect to $B$ must be greater than or equal to the given threshold, the scalars of membership degree of $A$ and $B$ must be greater than or equal to the given threshold, and the scalar of the membership degree of $A$ must be greater than or equal to the scalar of the membership degree of $B$.

In subclass–superclass hierarchies, a critical issue is multiple inheritance of class. Ambiguity arises when more than one of the superclasses have common attributes, and the subclass does not explicitly declare the class from which the attribute is inherited. Exactly which conflicting attribute in the superclasses is inherited by the subclass depends on their weights to the corresponding superclasses [21]. Also, it should be noted that in fuzzy multiple inheritance hierarchy, the subclass has different degrees with respect to different superclasses, not being the same as the situation in classical object-oriented databases [25].

![Fig. 5. A fuzzy generalization relationship in the fuzzy UML.](image-url)
In order to represent a fuzzy generalization relation, a dashed triangular arrowhead is applied. Fig. 5 shows a fuzzy generalization relationship. Here, classes Young Student and Young Faculty are all classes with the second level of fuzziness. That means that the classes may have some objects which belong to the classes with membership degree. These two classes can be generalized into class Youth, a class with the second level of fuzziness.

4.1.3. Fuzzy aggregation

An aggregation captures a whole-part relationship between a class named aggregate and a group of classes named constituent parts. The constituent parts can exist independently. Aggregate class Car, for example, is aggregated by constituent part classes Engine, Interior, and Chassis. Each object of an aggregate can be projected into a set of objects of constituent parts. Formally, let \( A \) be an aggregation of constituent parts \( B_1, B_2, \ldots, B_n \). For \( e \in A \), the projection of \( e \) to \( B_i \) is denoted by \( e \downarrow B_i \). Then we have

\[(e \downarrow B_1) \in B_1, (e \downarrow B_2) \in B_2, \ldots, \text{and } (e \downarrow B_n) \in B_n.\]

A class aggregated from fuzzy constituent parts must be fuzzy. If the former is still called an aggregate, the aggregation is fuzzy. At this point, a class is an aggregation of constituent parts with membership degree of \([0,1]\). We have the following criteria to determine the fuzzy aggregation relationship:

(a) For any (fuzzy) object, say \( e \), let the membership degree that it belongs to the aggregate, say \( A \), be \( \mu_A(e) \).

Also, let the projections of \( e \) to the constituent parts, say \( B_1, B_2, \ldots, B_n \), be \( e \downarrow B_1, e \downarrow B_2, \ldots, e \downarrow B_n \).

Let the membership degrees that these projections belong to \( B_1, B_2, \ldots, B_n \) be \( \mu_{B_1}(e \downarrow B_1), \mu_{B_2}(e \downarrow B_2), \ldots, \mu_{B_n}(e \downarrow B_n) \), respectively. Then \( \mu_A(e) = \min(\mu_{B_1}(e \downarrow B_1), \mu_{B_2}(e \downarrow B_2), \ldots, \mu_{B_n}(e \downarrow B_n)) \).

(b) Assume that a threshold, say \( \beta \), is given. Then \( \mu_A(e) \geq \beta \). Here, \( A, e, \) and \( \mu_A(e) \) are the same as the above.

Then \( A \) is the aggregate of \( B_1, B_2, \ldots, B_n \), with the membership degree \( \min(\mu_{B_1}(e \downarrow B_1), \mu_{B_2}(e \downarrow B_2), \ldots, \mu_{B_n}(e \downarrow B_n)) \). It is clear that \( \mu_A(e) \) cannot have a bigger value than any \( \mu_{B_1}(e \downarrow B_1), \mu_{B_2}(e \downarrow B_2), \ldots, \mu_{B_n}(e \downarrow B_n) \), and \( \mu_{B_i}(e \downarrow B_i) \). And \( \mu_{B_1}(e \downarrow B_1), \mu_{B_2}(e \downarrow B_2), \ldots, \mu_{B_n}(e \downarrow B_n) \) are not aggregated into 1 except that \( \mu_{B_1}(e \downarrow B_1), \mu_{B_2}(e \downarrow B_2), \ldots, \mu_{B_n}(e \downarrow B_n) \) are equal to 1. Formally, we have the following definition for the fuzzy aggregation relationship:

**Definition.** Let \( A \) be a fuzzy aggregation of fuzzy class sets \( B_1, B_2, \ldots, B_n \), which object membership functions are \( \mu_{A_k}, \mu_{B_1}, \mu_{B_2}, \ldots, \mu_{B_n} \), respectively. Let \( \beta \) be a given threshold. Then

\[(\forall e) (e \in A \land \beta \leq \mu_A(e) \leq min(\mu_{B_1}(e \downarrow B_1), \mu_{B_2}(e \downarrow B_2), \ldots, \mu_{B_n}(e \downarrow B_n))).\]

That means a fuzzy class \( A \) is the aggregate of a of group fuzzy classes \( B_1, B_2, \ldots, B_n \), if, for any (fuzzy) object instance, the scalar of the membership degree that it belongs to class \( A \) is less than or equal to the scalar of the member degree to which its projection to any of \( B_1, B_2, \ldots, B_n \), say \( B_i \), (1 \( \leq i \leq n \)), belongs to class \( B_i \). Meanwhile, for any (fuzzy) object instance, the scalar of the membership degree that it belongs to class \( A \) is greater than or equal to the given threshold.

Now let us consider the first level of fuzziness in the above-mentioned classes \( A, B_1, B_2, \ldots, B_n \), namely, they are the fuzzy classes with membership degrees.

**Definition.** Let \( A \) with degree \( A \), degree \( B_1 \) with degree \( B_1 \), degree \( B_2 \) with degree \( B_2 \), \ldots, degree \( B_n \) with degree \( B_n \) be classes. Here degree \( A \), degree \( B_1 \), degree \( B_2 \), \ldots, degree \( B_n \) are the scalars of membership degree. Let \( \mu_A, \mu_{B_1}, \mu_{B_2}, \ldots, \mu_{B_n} \) be the object membership functions of \( A, B_1, B_2, \ldots, B_n \), respectively. Then \( A \) is an aggregate of \( B_1, B_2, \ldots, B_n \) if

\[(\forall e) (e \in A \land \beta \leq \mu_A(e) \leq min(\mu_{B_1}(e \downarrow B_1), \mu_{B_2}(e \downarrow B_2), \ldots, \mu_{B_n}(e \downarrow B_n)) \land degree_A\]

\[\leq min(degree_{B_1}, degree_{B_2}, \ldots, degree_{B_n})].\]

Here \( \beta \) is a given threshold.
It should be noted that the assessment of fuzzy aggregation relationships given above is based on the extensional viewpoint of class. Clearly, these methods cannot be used if the classes are defined with the intensional viewpoint because there is no object available. In the following, we state how to determine the fuzzy aggregation relationship using the inclusion degree developed in [25].

**Definition.** Let $A$ be a fuzzy aggregation of fuzzy class sets $B_1, B_2, \ldots, B_n$, and $\beta$ be a given threshold. Also, let the projection of $A$ to $B_i$ be denoted by $A_{B_i}$. Then

$$\min(\mu(B_1, A_{B_1}), \mu(B_2, A_{B_2}), \ldots, \mu(B_n, A_{B_n})) \geq \beta.$$ 

Being the same as the fuzzy generation, here $\mu(B_i, A_{B_i})$ $(1 \leq i \leq n)$ means the membership degree to which $B_i$ semantically includes $A_{B_i}$. The membership degree that $A$ is an aggregation of $B_1, B_2, \ldots,$ and $B_n$ is $\min(\mu(B_1, A_{B_1}), \mu(B_2, A_{B_2}), \ldots, \mu(B_n, A_{B_n}))$.

Furthermore, the expression above can be extended for the situation that $A, B_1, B_2, \ldots,$ and $B_n$ have the first level of fuzziness, namely, they may be the fuzzy classes with membership degrees.

**Definition.** Let $\beta$ be a given threshold and $A$ WITH $\text{degree}_A$ DEGREE, $B_1$ WITH $\text{degree}_B_1$ DEGREE, $B_2$ WITH $\text{degree}_B_2$ DEGREE, $\ldots$ $B_n$ WITH $\text{degree}_B_n$ DEGREE be classes. Then $A$ is an aggregate of $B_1, B_2, \ldots,$ and $B_n$ if

$$\min(\mu(B_1, A_{B_1}), \mu(B_2, A_{B_2}), \ldots, \mu(B_n, A_{B_n})) \geq \beta \wedge \text{degree}_A \\
\leq \min(\text{degree}_{B_1}, \text{degree}_{B_2}, \ldots, \text{degree}_{B_n}).$$

A dashed open diamond is used to denote a fuzzy aggregate relationship. A fuzzy aggregation relationship is shown in Fig. 6. There, a car is aggregated by engine, interior, and chassis. The engine is old and we hereby have a fuzzy class *Old Engine* with the second level of fuzziness. Class *Old Car* aggregated by classes *interior* and *chassis* and fuzzy class *old engine* is a fuzzy one with the second level of fuzziness.

### 4.1.4. Fuzzy association

Two levels of fuzziness can be identified in the association relationship. The first level of fuzziness means that an association relationship fuzzily exists in two associated classes, namely, this association relationship occurs with a degree of possibility. Also, it is not known for certain if two class instances respectively belonging to the associated classes have the given association relationship, although this association relationship must occur in these two classes. This is the second level of fuzziness in the association relationship and is caused by such facts that

- the class instances belong to the given classes with possibility degree, and
- the role name of the association relationship is fuzzily defined, and there is a fuzzy association with membership degree between two connected instances (e.g., the degree of “friendship” of two people).

Also, it is possible that the two levels of fuzziness mentioned above may occur in association relationship simultaneously. This means that two classes have on one hand a fuzzy association relationship at class level. On the other hand, their class instances may have a fuzzy association relationship at the class instance level. Note that there may be one kind of fuzzy association modeled by linguistic labels. According to its semantics, the linguistic association should fall into one of the two levels of fuzziness in the fuzzy association relationship given above. Therefore, the linguistic association is not discussed in the paper.

---

Fig. 6. A fuzzy aggregation relationship in the fuzzy UML.
We can place a pair of words WITH *mem* DEGREE \((0 \leq mem \leq 1)\) after the role name of an association relationship to represent the first level of fuzziness in the association relationship. We use a double line with an arrowhead to denote the second level of fuzziness in the association relationship.

Fig. 7 shows two levels of fuzziness in fuzzy association relationships in the preliminary engineering design of a product car. Suppose that it is known that a part, a DVD player, may or may not be installed in the car at this stage. Assume the possibility that the DVD player will be installed in the car is 0.8. So in (a), classes *DVD Player* and *Car* have the association relationship installing with a 0.8 possibility degree. Also, it is possible that the part DVD player will certainly be installed in the car. Then the possibility that the DVD player will be installed in the car is 1.0 and classes *DVD Player* and *Car* have an association relationship installing with a 1.0 possibility degree. But as shown in (b), at the level of instances there exists the possibility that the instances of classes *DVD Player* and *Car* may or may not have an association relationship installing. In (c), two kinds of fuzzy association relationship in (a) and (b) arise simultaneously.

The classes with the second level of fuzziness generally result in the second level of fuzziness in the association if this association definitely exists (that means there is no first level of fuzziness in the association). Formally, let \(A\) and \(B\) be two classes with the second level of fuzziness, and have the object membership functions \(\mu_A\) and \(\mu_B\), respectively. Then the object instance \(e\) of \(A\) is one with a scalar of membership degree \(\mu_A(e)\), and the object instance \(f\) of \(B\) is one with a scalar of membership degrees \(\mu_B(f)\). Assume the association relationship between \(A\) and \(B\), denoted \(ass(A,B)\), is one without the first level of fuzziness. It is clear that the association relationship between \(e\) and \(f\), denoted \(ass(e,f)\), is one with the second level of fuzziness, i.e., with a scalar of membership degree, which can be calculated by

\[
\mu(ass(e,f)) = \min(\mu_A(e), \mu_B(f)).
\]

The first level of fuzziness in the association relationship can be indicated explicitly by the designers even if the corresponding classes are crisp. Assume that \(A\) and \(B\) are two crisp classes, and \(ass(A,B)\) is the association relationship with the first level of fuzziness, denoted \(ass(A,B)\) WITH *degree_ass* DEGREE. At this point, \(\mu_A(e) = 1.0\) and \(\mu_B(f) = 1.0\). Then

\[
\mu(ass(e,f)) = *degree_ass*
\]

The classes with the first level of fuzziness generally result in the first level of fuzziness of the association if this association is not indicated explicitly. Therefore, let \(A\) and \(B\) be two classes only with the first level of fuzziness, denoted \(A\) WITH *degree_A* DEGREE and \(B\) WITH *degree_B* DEGREE, respectively. Here *degree_A* and *degree_B* are the scalars of membership degree. Then the association relationship between \(A\) and \(B\), denoted \(ass(A,B)\), is one with the first level of fuzziness, namely, \(ass(A,B)\) WITH *degree_ass* DEGREE. Here *degree_ass* is a scalar and calculated by

\[
*degree_ass* = \min(*degree_A*, *degree_B*).
\]

For the instance \(e\) of \(A\) and the instance \(f\) of \(B\), in which \(\mu_A(e) = 1.0\) and \(\mu_B(f) = 1.0\), we have

\[
\mu(ass(e,f)) = *degree_ass* = \min(*degree_A*, *degree_B*).
\]

Finally, let us focus on the situation that the classes are ones with the first level and the second level of fuzziness, and there is an association relationship with the first level of fuzziness between these two classes, which is explicitly indicated. Let \(A\) and \(B\) be two classes with the first level of fuzziness, denoted \(A\) WITH *degree_A*

![Fig. 7. Fuzzy association relationships in the fuzzy UML.](image)
DEGREE and $B$ WITH $\text{degree}_B$ DEGREE, respectively. Let $\text{ass}(A, B)$ be the association relationship with the first level of fuzziness between $A$ and $B$, which is explicitly indicated with WITH $\text{degree}_\text{ass}$ DEGREE. Here $\text{degree}_A$, $\text{degree}_B$, and $\text{degree}_\text{ass}$ are the scalars of membership degree. Furthermore, let the object instance $e$ of $A$ be one with a scalar of membership degree $\mu_A(e)$, and the object instance $f$ of $B$ be one with a scalar of membership degree $\mu_B(f)$. Then we have

$$
\mu(\text{ass}(e, f)) = \min(\mu_A(e), \mu_B(f), \text{degree}_A, \text{degree}_B, \text{degree}_\text{ass}).
$$

4.1.5. Fuzzy dependency

The dependency between the source class and the target class is only related to the classes themselves and does not require a set of instances for its meaning. Therefore, the second level of fuzziness and the third level of fuzziness in class do not affect the dependency relationship. A fuzzy dependency relationship is a dependency relationship with degree of possibility, which can be indicated explicitly by the designers or be implied implicitly by the source class based on the fact that the target class is decided by the source class. Assume that the source class is a fuzzy one with the first level of fuzziness. Then the target class must be a fuzzy one with the first level of fuzziness. The degree of possibility that the target class is decided by the source class is the same as the possibility degree of source class.

Let $\text{Employee}$ and $\text{Employee Dependent}$ be two classes. It is clear that $\text{Employee Dependent}$ is dependent on $\text{Employee}$ and there is a dependency relationship between them. But it is possible that $\text{Employee}$ may have the first level of fuzziness, for example, with a 0.85 possibility degree. Correspondingly, $\text{Employee Dependent}$ also has the first level of fuzziness with a 0.85 possibility degree. Thus, the dependency relationship between $\text{Employee}$ and $\text{Employee Dependent}$ is a fuzzy one with 0.85 degree of possibility.

Since the fuzziness of the dependency relationship is denoted implicitly by first level of fuzziness of the source class, a dashed line with an arrowhead can still be used to denote the fuzziness in the dependency relationship. Fig. 8 shows a fuzzy dependency relationship.

In Fig. 9, we give a simple fuzzy UML data model utilizing some notations introduced in this chapter. Class $\text{Car}$ is a superclass, and $\text{New Car}$ and $\text{Old Car}$ are its two fuzzy subclasses, namely, they may have fuzzy instances. Similarly, class $\text{Employee}$ has three fuzzy subclasses: $\text{Young Employee}$, $\text{Middle Employee}$, and $\text{Old Employee}$. Classes $\text{Employee}$ and $\text{Car}$ have the fuzzy association relationship, using, which fuzziness is at the second level of fuzziness. Again, fuzzy classes $\text{Young Employee}$ and $\text{New Car}$ have a fuzzy association relationship, like, which fuzziness is at the first level of fuzziness. In addition, class $\text{Car}$ is aggregated by three classes: $\text{Engine}$, $\text{Chassis}$, and $\text{Interior}$. Class $\text{Engine}$ has three attributes. The attribute $\text{Id}$ and $\text{turbo}$ are ones with crisp values, whereas $\text{size}$ is a fuzzy attribute that can take fuzzy value. Classes $\text{Chassis}$ and $\text{Interior}$ are all crisp classes and they have no fuzziness at the three levels.

4.2. Transformation of the fuzzy UML to the fuzzy XML model

For our transformation approach, relevant constructs are the fuzzy extensions of those of UML’s Static View, consisting of the fuzzy classes and their relationship such as fuzzy association, fuzzy generalization, and various kinds of fuzzy dependencies. We develop the transformation of these constructs into DTD fragments.

4.2.1. Transformation of classes

UML classes are transformed into XML element type declarations [11]. Here, the class names become the names of the element types and the attributes are transformed into element content description. It is noted that, in the UML, attribute names are mandatory, whereas the attribute types are optional. In contrast, an element content only consists of type names in the XML. As a result, it is assumed that attribute names imply

```
Employee Dependent WITH 0.5 DEGREE

---------------

Employee WITH 0.5 DEGREE
```

Fig. 8. A fuzzy dependency relationship in the fuzzy UML.
their attribute type names [11]. When there is no class representing a suitable declaration for an attribute type, the attribute type is assumed to be an element whose content type is #PCDATA. In addition, multiplicity specifications of attributes are mapped into cardinality specifications with specifiers ?, *, and +, which are used for element content construction.

In the fuzzy UML model, four kinds of classes can be identified, which are

(a) classes without any fuzziness at the three levels,
(b) classes with fuzziness only at the third level,
(c) classes with fuzziness at the second level, and
(d) classes with fuzziness at the first level.

For the classes in case (a), they can be transformed following the approach developed in [11]. The transformation of the classes with the third and second levels of fuzziness is of particular concern. Instead of formal

```xml
<!ELEMENT e-mail (#PCDATA>).

<!ELEMENT student (sname?, age?, sex?, email?)>
<!ATTLIST student SID IDREF #REQUIRED>
<!ELEMENT sname (#PCDATA)>
<!ELEMENT age (Dist)>
<!ELEMENT Dist (Val+)>
<!ATTLIST Dist type (disjunctive)>
<!ELEMENT sex (#PCDATA)>
<!ELEMENT email (Dist)>
<!ELEMENT Dist (Val+)>
<!ATTLIST Dist type (conjunctive)>
<!ELEMENT Val (#PCDATA)>
<!ATTLIST Val Poss CDATA "1.0"/>

<!ELEMENT employee (Dist)>
<!ATTLIST employee FID IDREF #REQUIRED>
<!ELEMENT Dist (Val+)>
<!ATTLIST Dist type (disjunctive)>
<!ELEMENT Val (fname?, position?, office?, course?)>
<!ATTLIST Val Poss CDATA "1.0"/>
<!ELEMENT fname (#PCDATA)>
<!ELEMENT position (#PCDATA)>
<!ELEMENT office (#PCDATA)>
<!ELEMENT course (#PCDATA)>
```

Fig. 9. A fuzzy UML data model.

Fig. 10. Transformation of the classes in the fuzzy UML to the fuzzy XML.
definitions, in the following we utilize examples to illustrate how to transform the classes with the third and second levels of fuzziness into XML DTD.

First, let us look at class “student” in Fig. 10. It is clear that this class has two attributes, “age” and “e-mail”, taking fuzzy values representing various possible distributions. In other words, the class has the third level of fuzziness. While the class name becomes the name of the element type, and the attributes are transformed into element content description, these two attributes cannot be directly transformed into the element content description with content type `#PCDATA`. We should use

```xml
<!ELEMENT age (Dist)>
<!ELEMENT Dist (Val+)>
  <!ATTLIST Dist type (disjunctive)>
<!ELEMENT Val (#PCDATA)>
  <!ATTLIST Val Poss CDATA "1.0"/>
```

rather than use

```xml
<!ELEMENT age (#PCDATA)>.
```

Similarly, we use

```xml
<!ELEMENT email (Dist)>
<!ELEMENT Dist (Val+)>
  <!ATTLIST Dist type (conjunctive)>
<!ELEMENT Val (#PCDATA)>
  <!ATTLIST Val Poss CDATA "1.0"/>
```

in place of

```xml
<!ELEMENT e-mail (#PCDATA)>.
```

Now let us focus on class “employee” in Fig. 10. This class has the second level of fuzziness. That means that the class instances belong to the class with membership degrees. For such a class, when its class name becomes the name of the element type, the attributes cannot be transformed into element content description directly. We should use

```xml
<!ELEMENT employee (Dist)>
  <!ATTLIST employee FID IDREF #REQUIRED>
<!ELEMENT Dist (Val+)>
  <!ATTLIST Dist type (disjunctive)>
<!ELEMENT Val (fname?, position?, office?, course?)>
  <!ATTLIST Val Poss CDATA "1.0"/>
```

rather than directly use

```xml
<!ELEMENT employee (fname?, position?, office?, course?)>
  <!ATTLIST employee FID IDREF #REQUIRED>.
```

Fig. 10 depicts the details transforming classes “student” and “employee” into the fuzzy XML. An aggregation represents a whole-part relationship between an aggregate and a constituent part. We can treat all constituent parts as the special attributes of the aggregate. Then we can transform the aggregations using the approach to the transformation of classes.
4.2.2. Transformation of generalizations

The generalization in the UML defines a subclass/superclass relationship between classes: one class, called superclass, is a more general description of a set of other classes, called subclasses. Following the same transformation of classes given above, the superclass and each subclass are all transformed into the element types in the XML, respectively. Here the element type originating from the superclass is called a superelement and the element type originating from a subclass is called a subelement in [11]. Note that a superelement must receive an additional ID attribute stated #REQUIRED, and each subelement must be augmented by a #REQUIRED IDREF attribute in addition to the transformations that the class names become the names of the element types and the attributes are transformed into element content description.

Now consider the fuzziness in the generalization in the fuzzy UML model. Assume that the superclass and subclasses involved in the generalization may have fuzziness at the type/instance level (the second level) and/or at the attribute value level (the third level). The transformation of such superclass and subclasses can be finished according to the transformation of fuzzy classes developed above. Meanwhile, the created superelement and each subelement must be associated with ID #REQUIRED and IDREF #REQUIRED, respectively. Fig. 11 depicts the transformation of the fuzzy generalization.

4.2.3. Transformation of associations

Associations are relationships that describe connections among class instances. An association is a more general relationship than aggregation or generalization. So basically we can transform the associations in the UML using the approach to the transformation of generalizations given above. That is, first the class names become the names of the element types and the attributes are transformed into element content description. Then each element transformed must be augmented by a #REQUIRED IDREF attribute [15], which is an artificial one and from another class involved in the association.

Since, in the fuzzy UML data model, each class involved in an association may have fuzziness at the type/instance level (the second level) and/or at the attribute value level (the third level), its transformation must be carried out according to the transformation of fuzzy classes developed above. Utilizing this approach, Fig. 12 depicts the transformation of the fuzzy association.

5. Transformation of the fuzzy XML to the fuzzy relational model

This section presents the transformation of the fuzzy XML DTD to the fuzzy relational database model. Here we need a DTD tree created from the hierarchical XML DTD. The formal mapping from the fuzzy DTD tree to the fuzzy relational schema is developed in this section.

```xml
<!ELEMENT Young Faculty (age)>  
<!ATTLIST Young Faculty yfid IDREF REQUIRED>  
<!ELEMENT age (Dist)>  
<!ELEMENT Dist (Val+)>  
<!ATTLIST Dist type (disjunctive)>  
<!ELEMENT Val (name)>  
<!ATTLIST Val Poss CDATA “1.0”>  
<!ELEMENT name (#PCDATA)>  

<!ELEMENT Student (course)>  
<!ATTLIST Student sid IDREF #REQUIRED>  

<!ELEMENT Youth (Dist)>  
<!ATTLIST Youth yid ID #REQUIRED>  
<!ELEMENT Dist (Val+)>  
<!ATTLIST Dist type (disjunctive)>  
<!ELEMENT Val (name)>  
<!ATTLIST Val Poss CDATA “1.0”>  
<!ELEMENT name (#PCDATA)>  

Fig. 11. Transformation of the generalizations in the fuzzy UML to the fuzzy XML.
5.1. DTD tree and mapping to the relational database schema

The hierarchical XML and the flat relational data models are not fully compliant so the transformation is not a straightforward task. Generally, a DTD tree can be created from the hierarchical XML DTD. Its nodes are elements and attributes, in which each element appears exactly once in the graph, while attributes appear as many times as they appear in the DTD. The element nodes can be further classified into two kinds: leaf element nodes and nonleaf element nodes. So in the DTD tree, we have three kinds of nodes, which are attribute nodes, leaf element nodes, and nonleaf element nodes. Note that there exists a special nonleaf element node in the DTD tree, i.e., the root node. We also need to identify such attribute nodes that the corresponding attributes are associated with ID #REQUIRED or IDREF #REQUIRED in DTD. We call these attribute nodes key attribute nodes.

A DTD tree can be constructed when parsing the given DTD [40]. Fig. 13 shows a simple DTD tree example.

The created DTD tree is then mapped into the relational schema following the ensuing processing:

(a) Take the root node of the given DTD tree and create a relational table. Its attributes come from the attribute nodes and leaf element nodes connecting with the root node. Here the key attribute node(s) should become the primary key attribute(s) of the created table.
(b) For each nonleaf element node connecting with the root node, create a separate relational table. Its attributes come from the attribute nodes and leaf element nodes connecting with this nonleaf element node, and its primary key attribute(s) will come from the key attribute node(s).
(c) For other nonleaf element nodes in the DTD tree, apply the same processing given in (b) until all nonleaf element nodes are transformed.

Note that there may be cycles in DTD and element declarations that are referenced from more than one element declaration as element contents. Then we need to link a created relational table to its parent relational table through the parent table’s primary key.

The DTD tree in Fig. 13 is mapped into the relational schemas shown in Fig. 14.

5.2. Mapping the fuzzy XML model into the fuzzy relational database model

Generally speaking, the fuzzy XML DTD presented in Section 3 can be transformed into the fuzzy relational database schema using a similar processing as given above under a classical environment. That is, we first construct a DTD tree through parsing the given fuzzy DTD, and then map the DTD tree into the fuzzy relational database schema. However, the DTD tree here, called the fuzzy DTD tree, is clearly different from the classical DTD tree above because the fuzzy DTD contains new attribute and element types, which are attribute Poss and elements Val and Dist. As a result, the transformation of the fuzzy DTD tree to the fuzzy relational database schema is also different from the transformation of the classical DTD tree to the classical relational database schema.

In the fuzzy DTD tree, in addition to (key) attribute nodes, leaf element nodes, and nonleaf element nodes, there are three special nodes, which are Poss attribute nodes, Val element nodes, and Dist element nodes. Fig. 15 shows a simple fuzzy DTD tree that basically comes from the fuzzy DTD in Fig. 2. In this fuzzy DTD tree, the Dist element nodes created from Disk elements are used to indicate the type of a possibility distribution, being disjunctive or conjunctive. In addition, each Dist element node has a Val element node as its child node, and a nonleaf element node as its parent node.

From this figure, we can also identify four kinds of Val element nodes as follows:

(a) They do not have any child node except the Poss attribute nodes (type-1).
(b) They only have leaf element nodes as their child nodes except the Poss attribute nodes (type-2).
(c) They only have nonleaf element nodes as their child nodes except the Poss attribute nodes (type-3).
(d) They have leaf element nodes as well as nonleaf element nodes as their child nodes except the Poss attribute nodes (type-4).

In the following, we describe the transformation of the fuzzy DTD tree into the fuzzy relational database schema. Unlike the transformation of the classical DTD tree to the relational database schema, in the transformation of the fuzzy DTD tree to the fuzzy relational model the Poss attribute nodes, Val element nodes, and Dist element nodes in the fuzzy DTD tree do not take part in composing the created relational schema and only determine the model of the created fuzzy relational databases. To illustrate, we have the following process:

- university
  - UName
  - address
- employee
  - EID
  - ename
  - position
  - office
- department
  - DName
  - location
- student
  - SID
  - sname
  - sex
  - age

Fig. 14. The relational schema created by the DTD tree in Fig. 13.
(a) Take the root node of the given fuzzy DTD tree and create a relational table. Its attributes first come from the attribute nodes and leaf element nodes connecting with the root node. Here, the key attribute node(s) should become the primary key attribute(s) of the created table. Then determine if the root node has any Val element nodes or Dist element nodes as its child nodes. If yes, we need to further determine the type of each Val element node (we can ignore Dist element nodes because each Dist element node must have a Val element node as its child node only).

(i) If it is the Val element node of type-2, all of the leaf element nodes connecting with the Val element node become the attributes of the created relational table. An additional attribute is also added into the created relational table, representing the possibility degree of the tuples.

(ii) If it is the Val element node of type-3, only an additional attribute is added into the created relational table, representing the possibility degree of the tuples.

(iii) If it is the Val element node of type-4, we leave the nonleaf element nodes for further treatment in (b) and do the same thing as (i) for the leaf element nodes. It is impossible that the Val element nodes of type-1 arise in the root node.

(b) For each nonleaf element node connecting with the root node, create a separate relational table. Its attributes come from the attribute nodes and leaf element nodes connecting with this nonleaf element node, and its primary key attribute(s) will come from the key attribute node(s). Furthermore, determine if this nonleaf element node has any Val element nodes or Dist element nodes as its child nodes, and identify the type of these nodes, if any. We still apply the processing given in (i)–(iii) of (a) to treat the Val element nodes of type-2, type-3, and type-4. For the Val element nodes of type-1, each of them should become an
attribute of another relational table created from the parent node of the current nonleaf element. Note that this attribute is one that may take fuzzy values.

(c) For other nonleaf element nodes in the fuzzy DTD tree, apply the same processing given in (b) until all nonleaf element nodes are transformed.

Note that we could alternatively use a single relational table for some elements which are not the Dist, Vol, and leaf ones. In this case, all key attribute nodes from these elements should become the primary key attributes of the single table. Considering the high independence and easy maintenance of data in the table as well as the normalization of relational schema, we generally choose a relational table for one nonleaf element in the paper.

The fuzzy DTD tree in Fig. 15 is mapped into the fuzzy relational schemas shown in Fig. 16, in which attribute “age” is one that may take fuzzy values.

6. Conclusion

The wide utilization of the Web has resulted in the availability of huge amounts of electronic data. Information representation and exchange over the Web become important, and XML has been the de facto standard. On one hand, this creates a new set of data management requirements involving XML, such as the need to store and query XML documents. On the other hand, fuzzy sets and possibility distributions have been extensively applied to deal with information imprecision and uncertainty in real-world applications, and fuzzy database modeling is receiving increasing attention for intelligent data processing [23].

In order to manage fuzzy data in XML, this paper investigated the fuzzy XML data modeling. Based on possibility distribution theory, we first identified multiple granularity of data fuzziness in UML and XML. The fuzzy UML data model and fuzzy XML data model that address all types of fuzziness are developed. Further, we developed the formal conversions from the fuzzy UML model to the fuzzy XML model, and the formal mapping from the fuzzy XML model to the fuzzy relational databases. It should be noted that the fuzzy extension of XML in this paper only focuses on XML DTD because it has traditionally been the most common method for describing the structure of XML instance documents. But XML DTD lacks enough expressive power to properly describe highly structured data, and XML Schema provides a much richer set of structures, types and constraints for describing data [4]. The issues on fuzzy XML data modeling based on XML Schema will be stated in a forthcoming paper.

Acknowledgments

The authors thank the anonymous referees for their valuable comments and suggestions, which improved the technical content and the presentation of the paper. Many thanks are due to Lorna Tucci for reading our paper and offering editorial advice. Work is supported by the Program for New Century Excellent Talents in University (NCET-05-0288) and in part by the MOE Funds for Doctoral Programs (20050145024) and SRF for ROCS, MOE.

References


Z.M. Ma (Zongmin Ma) is currently a full professor at the Northeastern University, China. He received a Ph.D. from the City University of Hong Kong (China) in 2001. His research interests include database modeling, XML data management, as well as knowledge representation and reasoning with a special focus on information imprecision and uncertainty. He has published many papers in high quality international conferences (e.g., ER) and in highly cited international journals (e.g., Information Systems). He has also published two monographs with Springer. He has served as a Program Committee member of several conferences and workshops and has also spent some time as a reviewer of several journals such as IEEE Computer, Journal of Database Management, Knowledge-Based Systems, Knowledge and Information Systems, Fuzzy Sets and Systems, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, Soft Computing and Journal of Digital Information Management. Contact him at zongmin_ma@yahoo.com or mazongmin@ise.neu.edu.cn.

Li Yan is an associated professor at the Northeastern University, China. Her research interests include database modeling, XML data management, as well as imprecise and uncertain data processing. She has published papers in several journals such as Information and Software Technology and International Journal of Intelligent Systems.