ELSEVIER

Information Sciences 118 (1999) 101–119

INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL

www.elsevier.com/locate/ins

Modeling imprecise requirements with fuzzy objects

Jonathan Lee *, Nien-Lin Xue, Kuo-Hsun Hsu, Stephen J. Yang

Software Engineering Laboratory, Department of Computer Science and Information Engineering, National Central University, Chungli, Taiwan 32054

Received 1 January 1998; received in revised form 13 March 1999; accepted 13 April 1999

Abstract

One of the foci of the recent development in object-oriented modeling (OOM) has been the extension of OOM to fuzzy logic to capture and analyze informal requirements that are imprecise in nature. In this paper, a new approach to object-oriented modeling based on fuzzy logic is proposed to formulate imprecise requirements along four dimensions: (1) to extend a class by grouping objects with similar properties into a fuzzy class, (2) to encapsulate fuzzy rules in a fuzzy class to describe the relationship between attributes, (3) to evaluate the membership function of a fuzzy class by considering both static and dynamic properties, and (4) to model uncertain fuzzy associations between classes. The proposed approach is illustrated using the problem domain of a meeting scheduler system.

Keywords: Imprecise requirements; Object-oriented modeling; Requirements engineering

1. Introduction

One of the foci of the recent developments in object-oriented modeling (OOM) has been the extension of OOM to fuzzy logic to capture informal

* Corresponding author. Fax: +886-3 422 2681.

E-mail addresses: yjlee@se01.csie.ncu.edu.tw (J. Lee), nien@se01.csie.ncu.edu.tw (N.-L. Xue), glenn@se01.csie.ncu.edu.tw (K.-H. Hsu), jhyang@se01.csie.ncu.edu.tw (S.J. Yang)

0020-0255/99/$ - see front matter
PII: S0020-0255(99)00042-0

requirements that are imprecise in nature. Rumbaugh and his colleagues [27] have argued that OOM is a way of thinking about problems using models organized around real-world concepts which are usually expressed in natural languages. As Zadeh pointed out in [32], it is evident that almost all concepts in or about natural languages are almost fuzzy in nature. Several researchers such as Dubois et al. [12] and Lano [22] have further advocated that object classes with fuzzy memberships values are therefore a natural representation framework for real-world concepts.

As a continuation of our previous work [23,24] in using fuzzy logic as a basis for formulating imprecise requirements, we propose, in the paper, a fuzzy object-oriented modeling technique (FOOM) to capture and analyze imprecise requirements through the following two steps: (1) to identify the possible types of fuzziness involved in the modeling of imprecise requirements, and (2) to investigate the potential impacts of incorporating the notion of fuzziness on the features of object orientation.

In FOOM, we have identified several kinds of fuzziness that are required to model imprecise information involved in user requirements.

- classes with imprecise boundary to describe a group of objects with similar attributes, similar operations and similar relationships;
- rules with linguistic terms that are encapsulated in a class to describe the relationships between attributes;
- ranges of an attribute with linguistic values or typical values in a class to define the set of allowed values that instances of that class may take for the attribute;
- the membership degree (i.e. ISA degree) between an object and a class, and between a subclass and its superclass (i.e. AKO degree) can be mapped to the interval [0,1] and
- associations between classes that an object instance may participate to some extent.

The paper is organized as follows. We first introduce different versions of features in object orientation in the next section. Several kinds of fuzziness rooted in FOOM are fully described in Section 3. Related work is discussed in Section 4. Finally, we summarize the benefits of our approach and outline the plan for our future research in Section 5.

2. Features of object orientation

Object orientation is an engineering principle used to create a representation of the real-world problem domain and to map it into a software solution domain [13]. Different versions about the features of object orientation have been identified, for example, (1) Coad and Yourdon [9]: abstraction, information hiding, inheritance, and methods of organization including objects and attributes, assembly structures, and classification structures; (2) Rumbaugh

et al. [27]: identity, classification, polymorphism and inheritance; (3) Booch [5]: abstraction, encapsulation, modularity and hierarchy; (4) Wirfs-Brock and Johnson [30]: abstraction, encapsulation, polymorphism, classification and inheritance; and (5) Blair [4]: identity, encapsulation, classification, flexible sharing and interpretation.

We have adopted Blair's version of object orientation for, as was pointed out in [13], the five dimensions help in defining a framework which captures all the encountered variations.

- *Identity*: The unique identification of every object is through the use of an object identifier. An object's uniqueness is modeled even though its description is not unique [21].
- *Encapsulation*: The grouping together of various properties with an identifiable object is usually referred to as the notion of encapsulation. The main considerations in implementing encapsulation are [13]: (1) the selection of properties to be encapsulated (e.g. attributes, operations, representations, algorithms, triggers, constraints, etc.), and (2) the determination of visibility of these properties.
- *Classification*: The concept of classification is grouping associated objects according to common properties. Various classification schemes are possible such as the set, type, and class. It is important to distinguish between the intentional and the extensional aspects of a particular classification [13]:
  - The intent of a classification is the description of that classification characteristics, therefore, the intent specifies a predicate.
  - The extent of a classification is the set of objects in the current environment which features such behavior, that is, a population selected by applying a predicate.
- *Flexible sharing*: The ability for an object to belong to more than one classification is called flexible sharing, which will require a flexible form of behavior sharing. Techniques in achieving flexible sharing are based on the encapsulated properties, the considered classification scheme, and the relationships between classifications.
- *Interpretation*: The resolution of the precise semantics of the shared item of behavior. It is necessary to resolve the ambiguity exists in a flexible sharing environment. There are two steps involved in the resolution: type checking and binding. Type checking is the determination of whether operations are supported by a particular object; whereas, binding locates the correct implementation of the operation. However, interpretation is a feature that is more in line with the object-oriented programming aspect. Therefore, we will only focus on the other four features in FOOM.

In the next section, the four kinds of fuzziness in FOOM are fully discussed. Features of object orientation exhibited in FOOM are also described. The notations used throughout the paper is an extension of Unified Modeling Language (UML) [15].

3. Fuzzy object-oriented modeling

Fuzzy object-oriented modeling technique is a modeling approach for requirements engineers to model and analyze imprecise requirements. FOOM extends the traditional OOM along several dimensions: (1) to extend a class to a fuzzy class which classifies objects with similar properties, (2) to encapsulate fuzzy rules in a class to describe the relationship between attributes, (3) to evaluate fuzzy class memberships by considering both static and dynamic properties, and (4) to model uncertain fuzzy associations between classes.

3.1. Inside a fuzzy class

Traditionally, a class is used to describe a crisp set of objects with common attributes, common operations and common relationships. In order to model the impreciseness rooted in user requirements, we extend a class to describe a fuzzy set of objects (called a fuzzy class), in which objects may have similar attributes, similar operations and similar relationships, for example, a set of interesting books or a class of clever students. In the meeting scheduler system, the class *ImportantParticipant* is modeled as a fuzzy class, that is, a participant may be an important one to a degree.

A fuzzy class in FOOM is an encapsulation of a number of properties that can be classified as static properties or dynamic ones [1] (see Fig. 1). Static properties are viewed as integral features of an object that exist for its lifetime including identifier, attributes and operations. On the other hand, dynamic properties are optional for an object and can be short-lived such as fuzzy rules [2] and fuzzy relationships.

Since a fuzzy class is a group of objects with similar static properties (i.e., attributes, operations) and similar dynamic properties (i.e., relationships and rules), the membership degree of an instance to a fuzzy class is dependent on the properties, especially the values of attributes and the values of link attributes. In our example, the degree that a person belongs to the class *ImportantParticipant* depends on his *status* and his *role* in the meeting he attends.

The properties of fuzzy relationships and operations are discussed in detail in Sections 3.3 and 3.4 We focus on the fuzzy ranges and fuzzy rules in this section.

Attributes with fuzzy ranges: The *domain* of an attribute is the set of all values the attribute may take, irrespective to the class it falls into; whereas, the *range* of an attribute in a class is defined as the set of allowed values that

[1] We have adopted this classification scheme from Zenith approach [14].
[2] Encapsulating rules in an object is also proposed in SOMA [18].

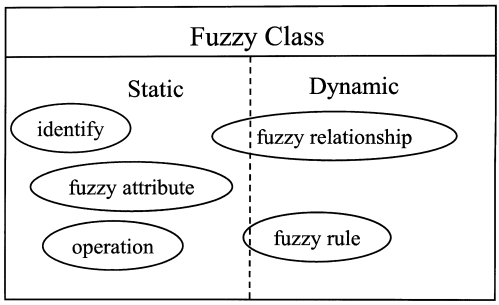

Fig. 1. Properties encapsulated in a fuzzy class.

a member of a class may take for the attribute [16]. The range of an attribute a_i in the class C is denoted as $R(a_i, C)$. In FOOM, the fuzziness in the range of an attribute in a class may be due to either a linguistic term or a typical value.

- A class may be fuzzy for the linguistic values its attributes can take. For example, the class *YoungMan* has a fuzzy range for the attribute *age*, since a person may take *young* or *very young* as values for his age.
- The range of an attribute is fuzzy because some of its values are deemed as atypical (i.e. less possible than other values), therefore, each value the attribute may take is associated with a typical degree [3]. In our example, the class *ImportantParticipant* has a fuzzy range {*student*/0.4, *staff*/0.7, *faculty*/1} for the attribute *status*, which means that a faculty is typically an important participant, and a student is an important participant with a typical degree of 0.4.

It is of interest to note that a crisp class may have attributes with fuzzy ranges. For instance, the class *MeetingRegistration* is a crisp class, with an attribute *participant importance*, which is associated with a fuzzy range.

Fuzzy rules: Incorporating fuzzy rules in object-oriented analysis can help enrich the semantics of analysis models. Using fuzzy rules is one way to deal with imprecision where a rule's conditional part and/or the conclusion part contains linguistic variables. More specifically, fuzzy rules in a fuzzy class play two important roles: to specify internal relationships between attributes, and to describe triggers more explicitly. These are usually neglected by most of the current object-oriented analysis approaches. Although Eckert and Golder [13] have addressed this issue by treating attributes as objects in which the internal relationships can be described, their approach ignores the fact that internal

[3] The notion of typical values is adopted from [12].

relationships not only exist in between attributes, but also among other encapsulated properties.

Fuzzy rules are an optional feature for a fuzzy class in FOOM and are thus classified as a dynamic property. Fuzzy rules are used to describe the internal relationship or external relationship. In the former, fuzzy rules describe the relationship between attributes inside a class. For example, in Fig. 2, a rule "if the *role* is a staff, the *participant importance* is *less important*" describes the relationship between the attributes *role* and *participant importance*. In the latter, fuzzy rules are used to describe the relationship between two different classes.

3.2. Fuzzy classification

Perceptual fuzziness refers to the compatibility between a class and an object (i.e. ISA), and the class membership between a class and its subclass (i.e. AKO) [31]. In FOOM, we extend crisp class memberships to fuzzy class memberships by allowing the existence of perceptual fuzziness. In this section, the notion of inheritance and how it effects the perceptual fuzziness are elaborated. A novel approach to calculating the membership degree of an object to a class and a subclass to its superclass is also proposed.

3.2.1. Subclassing and subtyping

Inheritance plays an important role in object-oriented analysis and design. Generally, inheritance is separated into two different concepts: implementa-

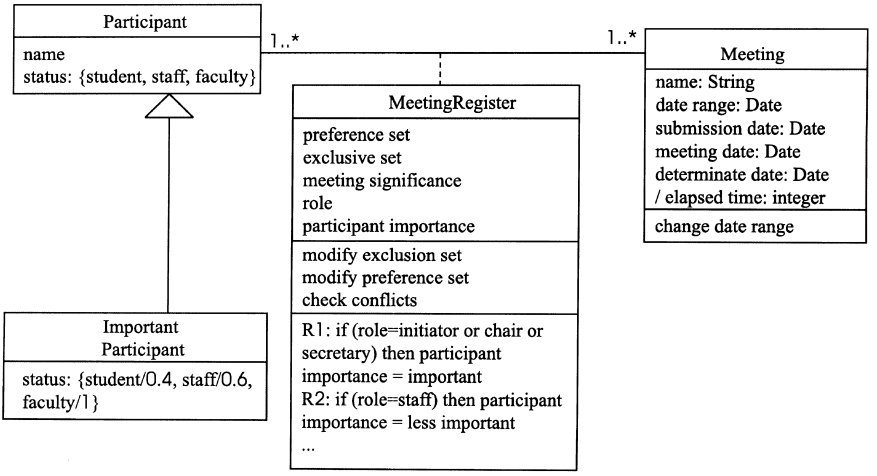

Fig. 2. An example of a fuzzy class.

tion inheritance and subtyping inheritance [8]. Implementation inheritance refers to the sharing of representation and implementation code between a subclass and its superclass. In contrast, subtyping refers to some kind of conformance between a subclass and its superclass in terms of their interfaces.

Subclassing is concerned with how a class is implemented, that is, a new class is constructed from a parent class by reusing some or all of the parent's operations [3]. In general, a subclass can be constructed through:

- *Extension*: add a new operation.
- *Redefinition*: retain the original interface but recode a particular method.
- *Restriction*: inherit a subset of operations.

On the other hand, a type defines an abstract interface and can be viewed as the specification of objects behavior [1,13]. There are three kinds of typing relationship: subtype, same type, and supertype. In the specification level, the subtype–supertype relationship can be defined in terms of the weak form of the principle of substitutability [8]:

> S is a subtype of T if substituting an object of type S wherever an object of type T is expected does not introduce the possibility of type errors.

To guarantee the weak form of the principle of substitutability, the following syntactic conditions are imposed.

- The subtype must provide at least all operations of its supertype.
- For each operation in the subtype, the corresponding operation in the superclass has the same number of arguments and results.
- The type of the arguments of the supertype's operations conform to those of the subtype's operations.
- The type of the result of the subtype's operations conforms to the type of the result of the supertype's operations.

It is well known that subclassing and subtyping are not necessarily related [3,13], namely, a subclass may be of different typing relationships: subtype, same type or supertype, to its superclass. If the subclass does not inherit the behavior of its superclass, a problem may occur: when an instance of the subclass is passed to a subprogram expecting an object of the superclass, the subprogram may produce unexpected results. In FOOM, a subclass is guaranteed to be either a subtype or a same type of its superclass, that is, the weak form of the principle of substitutability is maintained.

3.2.2. Perceptual fuzziness between classes and subclasses

Traditionally, the AKO relationship between a class and its subclass is crisp, that is, an instance of a subclass is also an instance of its superclass. In FOOM, an instance of the subclass may be an instance of its superclass to some extent, i.e., the AKO degree ranges from 0 to 1.

As the weak form of substitutability is maintained in FOOM, a subclass is constructed through extending new operations, redefining the inherited operations, adding new attributes or modifying the inherited attribute ranges. The AKO degree between a subclass and its superclass can be determined by examining the following situations:

- *Extension of operations and attributes*: In this case, an instance of the subclass carries all properties of its superclass, and is seen as an instance of its superclass. The AKO degree between the classes is equal to 1.
- *Redefinition of the inherited operations*: In this case, only implementation of the operation is redefined, an instance of the subclass still possesses the behavior of its superclass. Therefore, an instance of the subclass is viewed as an instance of its superclass.
- *Modification of the inherited attribute ranges*: In FOOM, since the attribute range in the superclass and subclass are allowed to be fuzzy, the attribute range in the superclass may include the attribute range in the subclass to some extent. In this case, an instance of the subclass is an instance of its superclass to a degree.

The perceptual fuzziness of an object to a class or a subclass to its superclass are calculated by evaluating both the static properties and dynamic properties. It is also important to note that not all attributes are necessarily related to the perceptual fuzziness. Referring to our example, the membership degree of a person to the class *ImportantParticipant* can be obtained by checking his *status* and his *participant importance* in the meeting he attends. Other properties such as his address, preference set or exclusive set are irrelevant to the perceptual fuzziness. An attribute that may affect a perceptual fuzziness is called a *Focus Of Attention* (FOA) *attribute*. Therefore, the attribute *status* is classified as a static FOA attribute, and *participant importance* a dynamic FOA attribute.

The criticality of an FOA attribute indicates the relevance of the attribute to a perceptual fuzziness. For example, the attribute *participant importance* is more relevant than the attribute *status* while examining if a participant is an important one, therefore, it is assigned a higher criticality. To determine the criticality of attributes, we utilize the Analytic Hierarchy Process (AHP) [28], which compares pairwise attributes according to their relative criticality. We use $\mathrm{CRI}(a_i, C)$ to denote the criticality of an FOA attribute a_i to the perceptual fuzziness for the class C. After the process of AHP, criticalities of attributes are normalized, that is, $\sum_{a_i \in \mathrm{FOA}(C)} \mathrm{CRI}(a_i, C) = 1$, where $\mathrm{FOA}(C)$ is the set of FOA attributes for the class C.

The membership degree of a class in its superclass has to account for the criticalities of the FOA attributes and the degree of inclusion of the range of each FOA attributes of the class in that of its superclass. Supposed that the class C is a superclass of the class D. The membership degree of the class C in the class D is denoted as $\mathrm{AKO}(D,C)$, and defined as

$$\mathrm{AKO}(D,C) = \sum_{a_i \in \mathrm{Att}(C) \cap \mathrm{FOA}(C)} \mathrm{CRI}(a_i, C) \times \mathrm{AKO}_{a_i}(D,C) + \sum_{E_k \in A(C)} \left(\sum_{b_j \in \mathrm{Att}(\langle C, E_k \rangle) \cap \mathrm{FOA}(C)} \mathrm{CRI}(b_j, C) \times \mathrm{AKO}_{b_j}(D,C) \right),$$

where $A(C)$ is the set of classes associated with C, $\mathrm{Att}(C)$ is the set of attributes in C, and $\mathrm{Att}(\langle C, E_k \rangle)$ is the set of link attributes in the association $\langle C, E_k \rangle$ which is established between the classes C and E_k. The degree $\mathrm{AKO}_{a_i}(x, C)$ is the AKO degree with respect to (w.r.t.) the attribute a_i and $\mathrm{AKO}_{b_j}(x, C)$ refers to the AKO degree w.r.t. to the link attribute b_j.

To calculate $\mathrm{AKO}a_i(D, C)$, we also need to examine whether the fuzziness of $R(a_i, C)$ is of the type of linguistic terms or typical values.

- In the case of linguistic terms, the membership degree is defined as the fuzzy inclusion of $R(a_i, D)$ into $R(a_i, C)$:

  $$\mathrm{AKO}_{a_i}(D,C) = \mathrm{INC}_I(R(a_i,C)|R(a_i,D)).$$

  The degree of inclusion of fuzzy sets is defined as

  $$\mathrm{INC}_I(A|B) = \frac{\|A \cap B\|}{\|B\|}.$$

  Note that $\mathrm{INC}_I(A|B) = 1$ if and only if $A \supseteq B$, and $\mathrm{INC}_I(A|B) = 0$ if and only if $A \cap B = \emptyset$. Therefore, $\mathrm{ISA}_{a_i}(D, C) = 1$ if and only if the range of a_i in the class D is included in the range of a_i in the class C, and $\mathrm{ISA}_{a_i}(D, C) = 0$ if there is no intersection in between.
- In the case of typical values, the membership degree is defined by a fuzzy inclusion based on Godel's fuzzy implication [10]:

  $$\mathrm{AKO}_{a_i}(D,C) = \mathrm{INC}_G(R(a_i,C)|R(a_i,D)).$$

The fuzzy inclusion $\mathrm{INC}_G(A|B)$ between the fuzzy sets A and B is defined as:

$$\mathrm{INC}_G(A|B) = \inf_s(\mu_B(s) \to \mu_A(s)), \text{where } \mu_B(s) \to \mu_A(s) = \begin{cases} 1 & \text{if } \mu_B(s) \leqslant \mu_A(s), \\ \mu_A(s) & \text{otherwise.} \end{cases}$$

Note that $\mathrm{INC}(R(a_i,C)|R(a_i,D)) = 1 \iff \forall s,\ \mu_R(a_i,D)(s) \leqslant \mu_R(a_i,C)(s)$, that is, the range of the attribute a_i in the subclass is included in the corresponding range of its superclass. More specifically, the subclass inherits attributes from its parents by specializing their ranges.

It is also required to examine whether the fuzziness of $R(b_i, \langle C, E_k \rangle)$ is of the type of linguistic fuzziness or typical values while calculating $\mathrm{AKO}_{b_j}(D, C)$:

- In the case of linguistic terms, the membership is defined as the fuzzy inclusion of $R(b_j, \langle D, E_k \rangle)$ into $R(b_j, \langle C, E_k \rangle)$:

$$\mathrm{AKO}_{b_j}(D, C) = \mathrm{INC}_I(R(b_j, \langle C, E_k \rangle) | R(b_j, \langle D, E_k \rangle)).$$

- In the case of typical values, we have

$$\mathrm{AKO}_{b_j}(D, C) = \mathrm{INC}_G(R(b_j, \langle C, E_k \rangle) | R(b_j, \langle D, E_k \rangle)).$$

3.2.3. Perceptual fuzziness between classes and objects

The class membership between an object and a class is crisp, that is, the ISA degree of an object to a class is either 1 or 0. In FOOM, a perceptual fuzziness between an object and a class is allowed. An object may belong to a class to a degree. In the meeting scheduler system, a person may belong to the class *ImportantParticipant* to some extent.

The ISA degree between an object and a class can be established either explicitly or implicitly [3]. In [17], the ISA degree is explicitly given, and used to derive the values of attributes of the object. Therefore, an object shares part of properties from its base classes. In FOOM, the ISA degree is implicitly determined by the structure of classes (see Fig. 3).

The membership degree of an object x in a fuzzy class C is denoted as $\mathrm{ISA}(x, C)$, and defined as:

$$\mathrm{ISA}(x, C) = \sum_{a_i \in \mathrm{Att}(C) \cap \mathrm{FOA}(C)} \mathrm{CRI}(a_i, C) \times \mathrm{ISA}_{a_i}(x, C) + \sum_{E_k \in A(C)} \left(\sum_{b_j \in \mathrm{Att}(C, E_k) \cap \mathrm{FOA}(C)} \mathrm{CRI}(b_j, C) \times \mathrm{ISA}_{b_j}(x, C) \right),$$

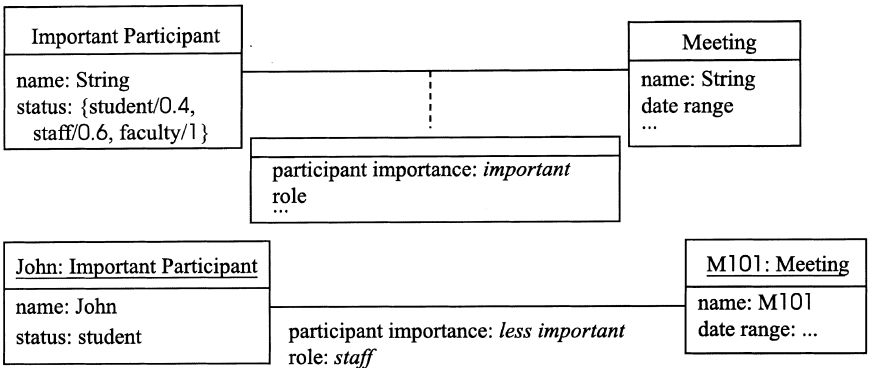

Fig. 3. An example of perceptual fuzziness for ISA relationship.

where $\mathrm{ISA}_{a_i}(x, C)$ is the membership degree of the object x in the class C with respect to (w.r.t.) the attribute a_i, and $\mathrm{ISA}_{b_j}(x, C)$ refers to the membership degree w.r.t. to the link attribute b_j. Both static properties (object attributes) and dynamic properties (link attributes) are required for computing the membership degree of an object in a class.

To calculate $\mathrm{ISA}_{a_i}(x, C)$, we need to examine whether the fuzziness of $R(a_i, C)$ is of the type of linguistic terms or typical values.

- In the case of linguistic terms, the membership degree is defined as the degree of inclusion of the value of a_i of x into the range of a_i in C, that is

$$\mathrm{ISA}_{a_i}(x, C) = \mathrm{INC}_I(R(a_i, C)|V(a_i, x)),$$

  where $V(a_i, x)$ is the value of x for a_i.
- In the case of typical values, the membership degree is defined by $\mu_{R(a_i,C)}V(a_i, x)$, that is,

$$\mathrm{ISA}_{a_i}(x, C) = \mu_{R(a_i,C)}V(a_i, x).$$

Similarly, to calculate the membership degree w.r.t. to a link attribute, we also need to check to see whether the fuzziness of the range of the link attribute's values is of the type of linguistic terms or typical values.

- In the case of linguistic terms, the membership degree is defined as the degree of inclusion of the value that b_j takes for the link $\langle x, e\rangle$ into the range of b_j in $\langle C, E_k\rangle$, where e is an instance of E_k connected with the object x:

$$\mathrm{ISA}_{b_j}(x, C) = \mathrm{INC}_I(R(a_i, \langle C, E_k\rangle)|V(a_i, \langle x, e\rangle)).$$

- In the case of typical values, we have

$$\mathrm{ISA}_{b_j}(x, C) = \mu_{R(a_i,\langle C,E_k\rangle)}V(a_i, x).$$

In our example, supposed that we have the following information in the meeting scheduler system:

$R(status, ImportantParticipant) = \{student/0.4, staff/0.6, faculty/1.0\}$
$R(participant\ importance, \langle\ ImportantParticipant,\ Meeting\ \rangle) = important$
$V\ (status, John)\ =$"$student$"
$V\ (role, John)\ =$ "$staff$"
$V(participant\ importance, \langle\ John, M101\ \rangle) = less\ important$
$CRI(status, ImportantParticipant)\ =\ 0.3$
$CRI(participantimportance, ImportantParticipant)\ =\ 0.7$

The value V (*participant importance*, *John*) is derived by the rule "if the role is staff, the participant importance is *less important*". To calculate the membership degree of a participant *John* to the class *ImportantParticipant* (abbreviated as IP), we first calculate the membership degrees w.r.t the FOA attributes *status* and *participant important* (abbreviated as pi).

$$\mathrm{ISA}_{\mathrm{status}}(\mathrm{John},\ \mathrm{IP}) = \mu_R(\mathrm{status},\ \mathrm{IP})(\mathrm{student}) = 0.4; \text{and}$$

$$\mathrm{ISA}_{\mathrm{pi}}(\mathrm{John},\ \mathrm{IP}) = \mathrm{INC}(\mathrm{important} \mid \mathrm{less\ important}) = 0.6.$$

Therefore, we have

$$\mathrm{ISA}(\mathrm{John},\ \mathrm{IP}) = \mathrm{CRI}(\mathrm{status},\ \mathrm{IP}) \times \mathrm{ISA}_{\mathrm{status}}(x,\ \mathrm{IP}) + \mathrm{CRI}(\mathrm{pi},\ \mathrm{IP}) \times \mathrm{ISA}_{\mathrm{pi}}(x, \langle \mathrm{IP},\ \mathrm{Meeting} \rangle) = 0.54.$$

It should be noted that the importance degree of *John* is dynamically determined by the meeting he attends. The example above describes that: "John is a more or less important participant (since the degree is 0.54) for the meeting *M*101".

3.2.4. Discussion

The concept of fuzzy clustering is somewhat similar to fuzzy classification in our approach: an entity (or object) can belong to a cluster (or a class) to some extent.

The primary objective of clustering techniques is to partition a given data sets into clusters. The concept of fuzzy clustering has been applied to the area of software engineering to the analysis of software systems structure, such as automatic clustering of a large number of program modules [2,26]. Techniques used to measure the distance between data include fuzzy similarity measure and fuzzy inclusion measure.

In FOOM, we use fuzzy inclusion technique to compute the compatibility degree between a class and an object, and the class membership between a class and its subclass (i.e., perceptual fuzziness). Instead of using fuzzy inclusion technique to clustering data, our focus is to determine the compatibility between objects based on their properties (i.e. attributes, operation, and association). In Table 1, we compare differences between our approach and fuzzy clustering.

Table 1
A comparison of fuzzy clustering and FOOM

	Objective	Application in software engineering field	Measure
Fuzzy clustering	Data partition [10]	Clustering program module [2,26]	Similarity, inclusion measure [26]
Perceptual fuzziness	Compatibility between entities and concepts [31]	Class classification in FOOM	Inclusion measure in FOOM

3.3. Uncertain fuzzy associations

Links and associations are means for establishing relationship among objects and classes. A link is a physical or conceptual connection between object instances. For example, John *work-for* Simplex company. An association describes a group of links with common structure and common semantics. For example, a person *work-for* a company. In traditional object-oriented approaches, only crisp associations are introduced, namely, an object either participates in an association or not.

Usually, certain and precise knowledge about an association is not always available in the user requirements; furthermore, users' observations are sometimes uncertain and imprecise. Therefore, an adequate management of uncertainty and imprecision in the phase of requirements analysis is an important issue. The distinction between imprecise and uncertain information can be best explained by Dubois and Prade [11]: imprecision implies the absence of a sharp boundary of the value of an attribute; whereas, uncertainty is an indication of our reliance about the fuzzy information.

A uncertain fuzzy association is allowed in FOOM. The imprecision of an association implies that an object can participate in the association to some extent, whereas uncertainty is referred to the confidence degree about the association. To represent the imprecision of an association, a special link attribute is introduced in FOOM to indicate the intensity that objects participate in an association. Fuzzy truth value, such as true, *fairly true* and *very true*, is used to serve as the representation of uncertainty for its capability to express the possibility of the degree of truth.

A link between x and y which is an instance of the association R is represented as a canonical form in FOOM:

$$(\text{link attribute}, \langle x, y\rangle, \text{ degree of participation}, \tau),$$

where the first component of the quadruple is a link attribute of an association. The value that a link $\langle x, y\rangle$ takes for the link attribute is described in the *degree of participation*, which represents the degree that objects x and y participate in the association R. The value is a linguistic term such as *very high*, *high* or *low*. The fuzzy valuation τ is a confidence level of the fuzzy association, whose value is a fuzzy truth value.

For example, an important participant can identify his preference for locations and the intensity of his preference (see Fig. 4). Sometimes it is not certain whether a participant prefer a specific location or not. To model the relationship between important participants and locations to be an uncertain fuzzy association *prefer* will help the meeting scheduler system to resolve conflicts and make a most convenient schedule. A link attribute *preference* is associated with the association *prefer* to indicate the degree of preference. By stating that a link between *John* and $L102$ is (preference, $\langle John, L102\rangle$,

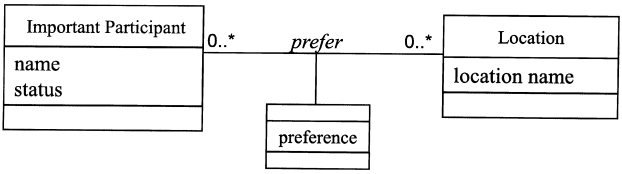

Fig. 4. An example of fuzzy association.

strong, *very true*), we mean that it is very true that *John* strongly prefers the location *L*102.

Constraints with imprecise information are also allowed in FOOM, called soft constraints. For example,the requirement "a meeting location should be as convenient as possible for all important participants" is modeled as a soft constraint on the associations *prefer* and *take place*.

3.4. Polymorphism

Polymorphism is one of the most characteristic features of object-oriented approaches. An operation may apply to many different classes is *polymorphic*, that is, the same operation takes on different forms in different classes. Several techniques have been proposed to support polymorphism, for example, overloading, coercions, parameter polymorphism, and inclusion polymorphism [3].

Inclusion polymorphism is adopted in FOOM. Inclusion polymorphism allows a function to operate on a range of types. The range of types is determined by *subtyping* relationships. With inclusion polymorphism, a function defined on a particular type can also operate on any subtype. Since FOOM maintains the weak form of the principle of substitutability, an operation defined in the superclass is viewed as a polymorphic operation.

4. Related work

A number of researchers have reported progress towards the successful integration of fuzzy logic and object-oriented modeling in the Fuzzy Logic and Software Engineering literature, e.g. [12,16,17,19,20,29], which can be classified into three categories based on their intended modeling purposes:

Knowledge representation for AI systems: Lano has proposed to combine fuzzy reasoning and object-oriented representation for the real-world information [22]. A knowledge base is organized as a class hierarchy for representing concept categories, each class corresponds to a fuzzy set, whose membership functions is the proximity metric defined for the class. Learning is the process that transforms a knowledge base and a new example to a new knowledge base.

To support an approximate reasoning in systems based on prototypical knowledge representation, Torasso and Console have defined a formalism for the representations and a general evaluation mechanism to deal with the form of knowledge [29]. Each frame has three kinds of weighted attributes: necessary, sufficient and supplementary. The evaluation mechanism is based on fuzzy logic: the fuzzy match between prototypical description and sets of data is based on possibility theory and the relevance measure of each slot.

To handle vagueness and imprecision in an expert system, Leung and Wong [25] integrate fuzzy concepts into object oriented knowledge representation. Relationships between classes may be crisp or fuzzy, which is dependent on the types of classes. If the relationship is fuzzy, the certainty factor of class variables and rules in a class will be modified in the subclass. An approach to querying fuzzy objects and the fuzzy relations between classes is also proposed.

Data modeling for database systems: In [12], Dubois and Prade have advocated that classes can be intensionally described in terms of attributes which are distinguished between the range of allowed values and the range of typical values. The degree of inclusion between a class C_1 and a subclass C_2 is computed by comparing the ranges or the typical ranges of C_1 with the ranges or the typical ranges of C_2. Three kinds of inheritance are proposed: typical inheritance, normal inheritance and atypical inheritance.

In [19], the problem of object recognition is viewed as a classification problem, which is characterized by an objected-oriented knowledge representation and control strategies based on fuzzy pattern matching procedures. Taxonomies of classes are represented by hierarchies of frames. Which class matching the unknown object is decided by fuzzy matching theory based on possibility theory. The matching process is seen as a quantification of similarity and differences of both objects, and the computation of these measures strongly depends on the types of the prototypes and object fields and the weight of each field.

Bodogna et al. [6] propose a Fuzzy Object Oriented model for management of crisp and fuzzy data. Their work develops a fuzzy graph-based data model, which intends to generalize a graph model so that imprecision and uncertainty can be managed at different levels. To formulate imprecise queries and retrieve precise or imprecise object with a degree of plausibility associated with them, fuzziness is managed in two level: imprecise in the data themselves and knowledge of the data, i.e., uncertainty in the information on the data.

Object-oriented modeling for conventional software systems: George et al. have utilized the ranges of fuzzy values of classes and objects for computing the degree of inclusion and membership, respectively [16]. To measure the class memberships, a similarity metric is formulated to measure the nearness between attributes' values in a superclass and its subclasses (see Fig. 5).

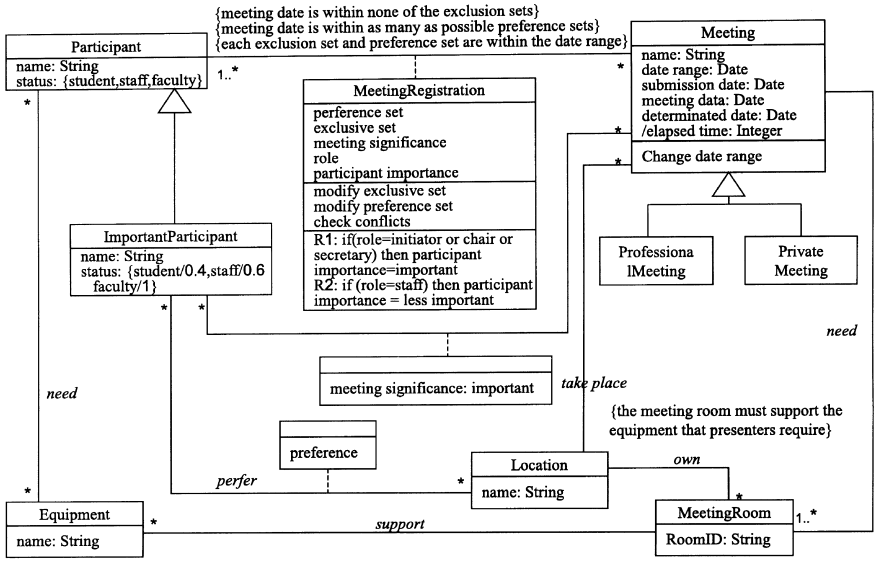

Fig. 5. Modeling meeting scheduler system using FOOM.

Graham [17] has focused on the derivation of unknown values of attributes through the use of a-kind-of relation (AKO), generalized modus ponen and defuzzification techniques [17]. In Graham's work, the notion of an object is extended to that of a fuzzy object in two ways: (1) attributes' values may be fuzzy, and (2) AKO is a matter of degrees. The AKO degree between classes is assumed to be known by a system, and unknown attributes' values are derived through AKO, generalized modus ponen and defuzzification techniques.

In [20], the focus has been on the representation of uncertain information based on a generalized fuzzy sets notation. Gyseghem et al. represent fuzzy information as fuzzy sets and uncertainty by means of generalized fuzzy set. A generic class Fuzzy-Set is introduced to capture fuzziness associated with attributes. Uncertain information is modeled by a kind of generalized fuzzy sets in which each element of the universe is associated with a fuzzy truth value $\{p/true, n/false\}$.

However, none of this work is devoted to the development of an object-oriented technique for informal requirements, but either to the formulation of class memberships (e.g. [12,17]), or to the representation of prototypical knowledge as in [29,19]. Furthermore, features of object orientation are not fully explored in these work. Types of fuzziness in consideration are also somewhat limited. Comparing FOOM with previous researches, FOOM offers several important benefits (see Table 2):

Table 2
Integrating OOM with fuzzy logic: a comparison

	Focus	Fuzziness			
		Attribute	Association	Classification	
				Property	Evaluation
Non-fuzzy technique	Modeling crisp requirements	Crisp value, crisp type	Certain, precise	Static, dynamic	Crisp
FOOM	Model imprecise requirements	Linguistic fuzziness, typical values	Degree of participation, uncertainty	Static, dynamic	Fuzzy implication
Bordogna et al. [6]	Imprecise data management	Type, class	Link strength, uncertainty	Static	Explicit
Dubois et al. [12]	Uncertainty in hierarchy	Possibility distribution	No	Static	Fuzzy implication
George et al. [16]	Imprecise data in data base	No	No	Static	Fuzzy similarity
Graham [17]	Fuzzy object	Linguistic fuzziness	No	Static	Explicit
Granger [19]	Object recognition	Uncertainty	No	Static	No
Leung and Wong [25]	Approximate reasoning	Linguistic fuzziness	No	Static	Explicit
Tarosso and Cambell [29]	Approximate reasoning	Possibility value	No	Static	No

- FOOM provides a more comprehensive modeling technique by considering different kinds of fuzziness that are usually found in user's requirements.
- These approaches only consider static attributes in computing the class membership, which neglects the fact that a fuzzy class is a fuzzy set of objects with similar attributes, operations and relationships. FOOM not only takes static attributes into account, but also relationships.
- FOOM is more general an approach than non-fuzzy ones in that FOOM can model both crisp and imprecise requirements. The benchmark against non-fuzzy modeling techniques can be evaluated based on the following aspects: focus of modeling, ranges of attributes in a class, association between classes, and the classification mechanism.

5. Conclusion

As was pointed by Borgida et al. [7], a good requirement modeling approach should take the problem of describing natural kinds into account; furthermore,

Zadeh have indicated that almost all concepts in or about natural languages are almost fuzzy in nature [32]. In this paper, we have proposed an approach to incorporating fuzzy concepts into object-oriented systems for modeling imprecise requirements. Several kinds of fuzziness involved in user requirements are identified: fuzzy classes, fuzzy rules, fuzzy ranges of attributes, perceptual fuzziness, and uncertain fuzzy associations.

Our approach offers an important benefit: to extend traditional object-oriented techniques to manage different kinds of fuzziness that are rooted in user requirements.

Our future research plan will consider the following tasks: (1) to explore the possibility of extending FOOM to handle trade-off analysis among conflicting requirements, and (2) to investigate the dynamic behavior in a fuzzy class.

Acknowledgements

This research was supported by National Science Council (Taiwan, Republic of China) under grants NSC87-2213-E-008-019.

References

[1] M. Ancona, Inheritance and subtyping, in: Proceedings of the Symposium on Applied Computing, 1991, pp. 382–388.
[2] L.A. Belady, C.J. Evangelisti, System partitioning and its measure, Journal of Systems and Software (1981) 23–29.
[3] B.S. Blair, Object-Oriented Languages, Systems, and Applications, Pitman, London, 1991.
[4] G. Blair, What are object-oriented systems? in: G. Blair et al. (Eds.), Object-Oriented Languages, Systems and Applications, Pitman, London, 1991, pp. 108–135.
[5] G. Booch, Object-Oriented Analysis and Design with Applications, Benjamin/Cummings, Menlo Park, CA, 1994.
[6] G. Bordogna, D. Lucarella, G. Pasi, A fuzzy object oriented data model, in: Proceedings of the Third IEEE Conference on Fuzzy Systems, 1994, pp. 313–318.
[7] A. Borgida, S. Greenspan, J. Mylopoulos, Knowledge representation as the basis for requirements specification, Computer (1985) 82–91.
[8] R.G. Clark, Type safety and behavior inheritance, Information and Software Technology 37 (10) (1995) 539–545.
[9] P. Coad, E. Yourdon, Object-Oriented Analysis, second ed., Springer, Berlin, 1990.
[10] D. Dubois, H. Prade, Fuzzy Sets and Systems: Theory and Applications, Academic Press, NewYork, NY, 1980.
[11] D. Dubois, H. Prade, Possibility Theory: An Approach to Computerized Processing of Uncertainty, Plenum Press, New York, NY, 1988.
[12] D. Dubois, H. Prade, J.P. Rossazza, Vagueness, typicality and uncertainty in class hierarchies, International Journal of Intelligent Systems 6 (1991) 161–183.
[13] G. Eckert, P. Golder, Improving object-oriented analysis, Information and Software Technology 36 (2) (1994) 67–86.

[14] Z.P. Kemp et al., Zenith system for object management in distributed multimedia design environments, Information and Software Technology 34 (7) (1992) 427–436.
[15] M. Fowler, K. Scott, UML Distilled: Applying the Standard Object Modeling Language, Addison-Wesley, Reading MA, 1997.
[16] R. George, R. Srikanth, F.E. Petry, B.P. Buckles, Uncertainty management issue in the object-oriented data model, IEEE Transitions on Fuzzy Systems 4 (2) (1996) 179–192.
[17] I. Graham, Fuzzy objects: inheritance under uncertainty, in: Object Oriented Methods, Addison-Wesley, Reading MA, 1994, pp. 403–433.
[18] I. Graham, Migration using SOMA: a semantically rich method of object-oriented analysis, Journal of Object-Oriented Programming (1993) 31–42.
[19] C. Granger, An application of possibility theory to object recognition, Fuzzy Sets and Systems 28 (3) (1988) 351–362.
[20] N.V. Gyseghem, R.D. Caluwe, R. Vandenberghe, UFO: uncertainty and fuzziness in an object-oriented model, in: Proceedings of the Internationl Conference on Fuzzy Systems, 1993, pp. 773–778.
[21] S.N. Khoshafian, G.P. Copeland, Object identity, in: Proceedings of the Conference on Object-Oriented Programming Systems, Languages and Applications, 1986, pp. 406–416.
[22] K. Lano, Combining object-oriented representations of knowledge with proximity to conceptual prototypes, in: Proceedings of Computer Systems and Software Engineering, 1992, pp. 442–446.
[23] J. Lee, J.Y. Kuo, Fuzzy decision making through trade-off analysis between criteria, Information Science: An International Journal 107 (1998) 107–126.
[24] J. Lee, J.Y. Kuo, New approach to requirements trade-off analysis for complex systems, IEEE Transactions on Knowledge and Data Engineering 10 (4) (1998).
[25] K.S. Leung, M.H. Wong, Fuzzy concepts in an object oriented expert system shell, International Journal of Intelligent Systems 7 (1992) 171–192.
[26] T. Raz, A.T. Yaung, Application of clustering techniques to information systems design, Information and Software Technology 37 (3) (1995) 145–154.
[27] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen, Object-Oriented Modeling and Design, Prentice-Hall, Englewood Cliffs, NJ, 1991.
[28] T.L. Saaty, The Analytic Hierarchy Process, McGraw-Hill, New York, NY, 1980.
[29] R.B. Terwilliger, R.H. Cambell, Please: executable specifications for incremental software develoment, The Journal of Systems Software 10 (1989) 97–112.
[30] R. Wirfs-Brock, B. Wilkerson, L. Wiener, Designing Object-Oriented Software, Prentice-Hall, Englewood Cliffs, NJ, 1990.
[31] V. Wuwongse, M. Manzano, Fuzzy conceptual graphs, in: G.W. Minean, B. Moulin, J.F. Sowa (Eds.), Conceptual Graphs for Knowledge Representation, 1993, pp. 430–449.
[32] L.A. Zadeh, Test-score semantics as a basis for a computational approach to the representation of meaning, Literacy Linguistic Computing 1 (1986) 24–35.