Extending UML 1.5 for fuzzy conceptual modeling: An strictly additive approach

MIGUEL-ANGEL SICILIA[1], NIKOS MASTORAKIS[2]
[1]Computer Science Department
University of Alcalá
Ctra. Barcelona, km.33,6 – 28871 Alcalá de Henares (Madrid)
SPAIN

[2]Military Inst. of University Education
Hellenic Naval Academy
Department of Electrical and Computer Engineering
Terma Hatzikyriakou, 18539, Piraeus
GREECE

Abstract: - The Unified Modeling Language (UML) has become a widespread notation for conceptual modeling, and it currently provides a number of extensibility mechanisms to tailor it to specialized modeling issues. Some extensions have been proposed to extend the UML for fuzzy modeling, but without explicitly considering strict additivity and semantic compatibility with the original specification. In this paper, an analysis of these two issues is provided, describing the type of extensions that can be integrated with the current specification without breaking its semantics. Concretely, some forms of strictly additive fuzzy classification, fuzzy association, fuzzy generalization and fuzzy attribute values are sketched.

Key-Words: - Fuzzy Conceptual Modeling, Unified Modeling Language, Meta-modeling, fuzzy class, fuzzy association, fuzzy generalization, fuzzy attribute values.

1 Introduction

The *Unified Modeling Language* (UML) [3] has become a widely accepted graphical notation for conceptual modeling, and a large number of Software Engineering and general-purpose modeling tools now provide support for it. The UML has evolved into an OMG standard[1], and its current 1.5 version is now subject of a major 2.0 revision that would eventually be available in 2004.

Nonetheless, despite the flexibility of the UML to be adapted to diverse specific usages (e.g. business modeling, data modeling, service description, etc.), the UML does not provide explicit means to deal with information imperfections like uncertainty and imprecision [10], which were somewhat addressed in previous conceptual notations like *Entity-Relationship* models [1], and that can be considered an important domain aspect [6]. This has lead to some previous attempts to extend the UML in several dimensions of fuzzy conceptual modeling [2, 7, 9]. But these proposed extensions did not deal with the issue of analyzing which extensions are possible to add without breaking in some way the original core semantics of the language, and therefore, without loosing compatibility with existing tools and systems supporting the UML. Such an analysis is required from the viewpoint that non-compliant extensions would likely not be easily introduced in standard UML tools, and they may also introduce inconsistencies in formal accounts of conceptual modeling.

This paper provides an analysis of the extensions that can be added to the UML 1.5 version in a *strictly additive* manner, i.e. those extensions that:

> (a) can be expressed through the built-in UML extensibility mechanisms, and that
> (b) do not break the semantics of the original model specification.

[1] http://www.omg.org/uml

This analysis is restricted in this paper to the UML static definition, i.e. it only addresses elements in the *Foundation* package of the UML, as a first natural step to address the behavioral elements that depend on it. In what follows, we will refer to "fuzziness" in a broad sense to refer to imprecision and uncertainty.

2 Background

The architecture of the UML language is based on a four meta-model structure with the following layers: user-objects (M_0), model (M_1), meta-model (M_2) and meta-metamodel (M_3). The language is described in terms of M_2, in a semi-formal way, by using three views, namely: *abstract syntax*, *well-formedness* rules and *semantics*.

The abstract syntax is expressed in terms of UML diagrams, rules are provided in Object Constraint Language (OCL) language, and semantics are provided in natural language (english). Nonetheless, OCL rules are not required to describe completely the semantics of the elements, and are actually used only to express a limited number of constraints. A UML *profile* is a coherent set of extensions to the UML aimed at a particular usage or domain. Figure 1 depicts the main packages in the UML M_2 level.

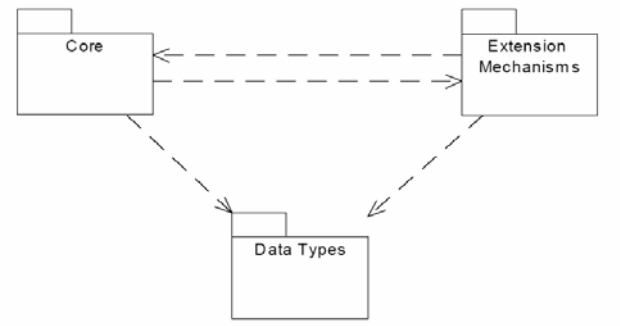

Figure 1: UML *Foundation* Packages at M_2.

The *Core* package at M_2 defines the basic modeling elements of the language, so that our analysis will essentially address the elements at this level. This is due to the fact that entities at this level actually provide the description of the elements used in tools. The *Data types* package provides primitive type definitions that are used to define the core elements.

As described elsewhere [1, 2], fuzziness can be introduced in modeling notations with two different and orthogonal intentions:

(i) As a mean to express fuzziness about the appropriateness of the modeling elements with respect to the domain being modeled. In this case, fuzziness comes from the modelers.

(ii) As a mean to express fuzziness about the entities described in the model. In this case, fuzziness comes from some aspects of the domain being modeled.

It should be noted that intention (i) models fuzziness related to the modeling process itself, while intention (ii) models fuzziness in the domain, so that they can be combined in a given model, but they do not interact. Intentions (i), and (ii) are addressed respectively in the following two sections.

3 Fuzziness in the modeling process itself

Fuzziness about the relationship between the elements in the model and the domain can be used to convey several kinds of information, including the following:

- It's not completely sure the role element *E* plays in the context of the model.
- There's a partial degree of the compatibility of the information that needs to be represented with what the model intends to describe.
- Element *E* is "candidate", in the sense that we're not sure about the appropriateness of its inclusion in the model or about its structure (its constituents, attributes, operations and the like).

It should be noted that this kind of fuzziness is common in modeling settings. For example, the concept of *candidate* classes or elements is common practice in object-oriented modeling methods as for example OMT [5].

The introduction of numerical or linguistic fuzzy handling at this level would require extensions to the model elements at level M_3, which the UML specification (p. 2-5) defines in the following way: "the MOF meta-metamodel is the meta-metamodel for the UML metamodel". In other words, the extensions should be carried out at the level of the meta-metamodel of the *Meta-Object-Facility* (MOF) specification [4].

Let's take the example of defining candidate classes with a level of uncertainty. A definition at level M_3 would look as sketched in the following specification fragment:

```
MetaClass("Candidate Class",
  [MetaAttribute("name", String),
   MetaAttribute("fields",
                 List<"Field">),
   MetaAttribute("candidate-fields",
            List<"CandidateField">)
   MetaAttribute("degree", [0,1])
...])
MetaClass("Candidate Field",
     [MetaAttribute("name", String),
      MetaAttribute("degree", [0,1]),
     ...]
)
```

Where the *MetaAttribute* "degree" is intended to hold the degree of "candidateness" of the class, with the zero value being absolute doubt about the appropriateness of its inclusion in the model, and the one value being absolute certainty.

Similar definitions could be used to describe candidate attributes or operations (like "Candidate Field" in the fragment above), and some well-formedness rules could be used to model specific numerical approaches for this kind of fuzziness in "generalization-specialization" relationships, like those described by Ma in [2].

Technically, this kind of definition of candidate elements does not require an extension to the MOF, so that this kind of fuzziness can be effectively modeled in the framework of MOF systems.

But even though the described semantics do not preclude doing such kind of extensions, the built-in mechanisms of UML are targeted to express specialized characteristics of model elements not concerned with the MOF, so that they can not be expressed as UML *profiles*, and thus can't be considered to conform to condition (a) described in the introduction to this paper.

4 Extensions in the modeled entities

Extensions applicable to entities at M_1 reflect in fact specificities of elements at M_0. For example, a "fuzzy class" at M_1 can be considered as a fuzzy set (collection) of instances at M_0, for which a degree of membership to the class is somewhat provided. This type of extensions can be defined by using the UML extension mechanisms mentioned before: stereotypes, constraints, tag definitions, and tagged values.

An stereotype introduces (by its mere definition) a new meta-class in UML by "subclassing" an existing one, i.e. it introduces a new modeling concept that shares all the previous properties of a previous one, and also introduces some new ones. In fact, the UML specification uses stereotypes to define some of its main elements, for example, the stereotype <<persistent>> can be attached to classes to denote that their instances are stored in secondary memory. The main advantage of stereotypes is that they can be flexibly defined by modelers at any type, and they allow tailoring visual presentation. This concept can be used to introduce fuzzy classes, fuzzy generalizations and fuzzy associations, which are among the most common kinds of conceptual modeling elements. The rest of this section discusses some of these elements.

4.1 Fuzzy classes

Table 1 defines a `FuzzyClass` as a specific type of class. Tags can be used to define additional characteristics of the fuzzy class. For example, the `isNormal` (which can be given a `UML::Datatypes::Boolean` type) tag can be used to indicate that the class is normal, i.e. that at least one objects has a full degree of membership. This definition by itself does not violate the semantics of the `Class` element in the UML. Some other fuzzy set related concepts could also be modeled this way, since fuzzy classes can be actually considered as fuzzy sets defined extensionally by their set of instances.

Stereotype	Base class	Tags
FuzzyClass	Class	isNormal

Table 1: A possible simple stereotype for fuzzy classes

Instantiation is also not problematic, since the UML allows for multiple classifications (i.e. an instance being part of more than one class). Nonetheless, fuzzy class definitions that allow the fact that an instance does not have all the properties of its subclasses would conflict with the strict instantiations of the UML, which precludes the use of fuzzy deductive capabilities that result in such definitions. This is related to the fact that in UML instances must conform to all of the properties declared by its class(es), which makes sense for imperative object-oriented programming, but not for some soft approaches to conceptual knowledge representation [8].

In terms of well-formedness, the fact that membership degrees should be stored for each instance can be expressed through a rule like the following one, that should be interpreted in the context of the extended class `FuzzyClass` at M_2:

```
self.feature->
  select (a| a.oclIsKindOf (Attribute)
   )->
    exists ( p | p.name = "degree" and
              p.type = UnitInterval)
```

The rule makes the modeling system ensure that a degree attribute exists for each fuzzy class, and that it stores normalized membership values (represented by the `UnitInterval` type).

4.2 Fuzzy associations

A fuzzy association can be defined in a similar way to a fuzzy class as showed in Table 2. In this case, `isTransitive` is a boolean property that can be used to model fuzzy instance relations like for example, similarity relations, which are inherently transitive [7].

Stereotype	Base class	Tags
FuzzyAssociation	Association	isTransitive

Table 2: A possible simple stereotype for fuzzy associations

A rule can be used to enforce the provision of a standardized attribute to hold association relation values, but this would require making `FuzzyAssociation` a stereotype of the `AssociationClass` instead of `Association`, since the former can have attributes according to the UML meta-model. Such rule can be similar to the one showed for classes, but it should be interpreted in the context of `FuzzyAssociation`:

```
self.feature->
select (a| a.oclIsKindOf
        (Attribute) )->
   exists (p|p.name = "strength"
   and p.type = UnitInterval)
```

Note that since `AssociationClass` is a `Class` as defined in the UML meta-model, a `FuzzyAssociation` will have two different interpretations of instance-memberships, one connected with the strength of the links and other with the relation to the association class by itself. This could be redundant in many applications, so that it would be desirable to provide a new tag to identify cases in which “strength equals to grade".

The above definition does not break current `Association` UML Semantics. Nonetheless, other kind of extensions can not be included. An important example is that of fuzzy multiplicities, denoting approximate ranges of permitted instances at an end of the association. The UML states that “A multiplicity is a range of nonnegative integers”, so that for example, fuzzy ranges represented as pairs of fuzzy numbers or linguistic labels (e.g. “many", “few"), could not be used without a formal violation of association semantics. Nonetheless, such kind of imprecision in cardinalities is commonplace in the course of creating a model.

4.3 Fuzzy generalizations

A fuzzy generalization can be defined as showed in Table 3.

Stereotype	Base class	Tags
FuzzyGenls	Generalization	delta, distance

Table 3: A possible simple stereotype for fuzzy generalizations

In this case, `delta` and `distance` are used to determine the *discriminant* of the specialization (i.e. the rationale for defining a subset of instances) and a possible distance for the super-class, which could be used to represent cognitive distances as those described in [8]. For example, the discriminant for the specialization of the class `Cat` to `SiameseCat` can be expressed as “race", while the discriminant from `Cat` to `WildCat` can be expressed as “habitat". In addition, it would be possible that the distance of the former be 0.6, while the distance to the latter be 0.8, indicating that the latter term has more differences with the superclass than the former.

Fuzzy subclassing entails that the instances in a (fuzzy) class *A* can not have greater membership values on superclasses of *A*, as defined in [1]. The contrary would result in the paradox that such instance is more closely connected to a more abstract concept than to a more specific one, which in addition contradicts the first case. Formally, if *B* is a subclass of *A*,

$$\mu_A(x) \leq \mu_B(x) \quad \forall A,B \in Class.$$

4.4 Attributes holding fuzzy values

Fuzzy values can be considered as generalizations of some primitive data types. For example, fuzzy numbers can be used to express imprecise integer or real quantities, or even ill-defined intervals. Such extensions could then be integrated just by the mechanism of stereotypes described so far. As the UML does not mandate a single data type hierarchy, new forms of data types can be added simply as special classes at the M_1 level. For example, if triangular fuzzy numbers are required, a `TriangularFuzzyNumber` element can be defined simply as a class, and be subsequently used in any part of the model under development, as depicted in Figure 2.

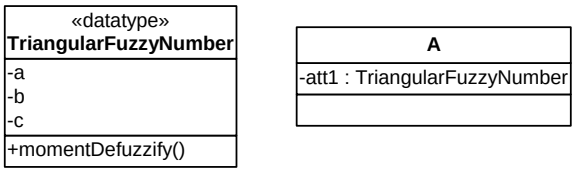

Figure 2: Example definition and usage of fuzzy data types.

Triangular fuzzy numbers can be defined by three real numbers *(a,b,c)* delimiting the shape of the triangle. In addition, convenience operations could be defined on the new type. For example, an operation for obtaining a crisp real number from the fuzzy number using the moment defuzzifier, as depicted in Figure 2. The <<datatype>> stereotype is used in the UML standard to differentiate data types from classes, since the former do not have a formal requirement of identifying uniquely their instances. Such an approach can also be used to introduce linguistic label sets (i.e. "low", "medium", "high"), as special kinds of enumerations, in which each label represents a fuzzy set.

The defined stereotypes are showed in UML compliant tools as depicted in Figure 3.

It should be noted that the UML allows for specialized visual representations of stereotypes to help model comprehensibility. In consequence, research on the usability (in the sense given in [9]) of the diverse alternatives should be carried out.

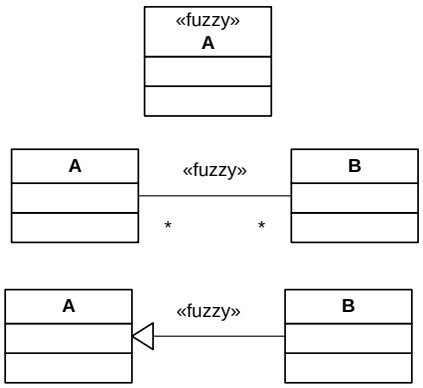

Figure 3: Standard visual notation for fuzzy stereotypes.

5 Conclusions and Future Work

The current meta-modeling architecture of the UML 1.5 language allows for diverse types of extensions that can be used to include fuzziness as an integral part of modeling. Fuzziness at the level of the relationship model-model entity can be described in terms of the MOF (the UML meta-metamodel), although this can not be properly considered as a UML extension, but an alternative, extended MOF-compliant meta-model.

Common fuzzy extensions dealing with imprecision and uncertainty at the instance or class levels can be accommodated just by the built-in UML extensibility mechanisms, carefully checking that existing semantics are not violated. Examples of those allowed extensions are fuzzy classification, fuzzy associations and fuzzy generalization.

Future work should revisit the extensions presented here with regards to the forthcoming new extensibility features that were announced for the UML 2.0 version, and also provide a more detailed account of the feasibility of accommodating existing proposed fuzzy extensions.

References:

[1] Chen, G., *Fuzzy Logic in Data Modelling: Semantics, Constraints, and Database Design*. The Kluwer International Series on Advances in Database Systems, Kluwer, 1998.

[2] Ma, Z. M., Extending UML for Fuzzy Information Modeling. In: *Object-Oriented Databases: Theories and Practices*, Idea Group Publishing, 2004 (in print).

[3] Object Management Group, *OMG Unified Modeling Language Specification*. March 2003 Version 1.5, formal/03-03-01.

[4] Object Management Group, *Meta Object Facility (MOF) Specification*, version 1.4, April 2002.

[5] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W. *Object-Oriented Modeling and Design*, Prentice Hall, Englewood Cliffs, NJ, USA, 1991.

[6] Sicilia, M.A. and García, E., On Imperfection in Information as an "Early" Cross-cutting Concern and its Mapping to Aspect-Oriented Design. In *Proceedings of the Early Aspects Workshop: Aspect-Oriented Requirements Engineering and Architecture Design*, Lancaster, UK, 2004.

[7] Sicilia, M.A., Gutiérrez, J.A., García, E., Designing Fuzzy Relations in Orthogonal Persistence Object-Oriented Database Engines. *Advances in Artificial Intelligence - IBERAMIA 2002*, Lecture Notes in Computer Science, 2527 Springer, 2002, pp. 243-253.

[8] Sicilia, M. A., García, E., Aedo, I., Díaz, P., Representation of Concept Specialization Distance through Resemblance Relations. In *Advances in Soft Computing - Engineering, Design and Manufacturing* (Springer Engineering series). Springer Verlag, 2003, pp. 173-182.

[9] Sicilia, M.A., García, E., Gutiérrez, J.A., Introducing Fuzziness in Existing Orthogonal Persistence Interfaces and Systems. In: *Advances in Fuzzy Object-Oriented Databases: Modeling and Applications*, IDEA Group Publishing, 2004 (to appear).

[10] Smets, P., Imperfect information: Imprecision-Uncertainty. In Motro, A. Smets, P. (eds.): *Uncertainty Management in Information Systems: From Needs to Solutions*, Kluwer Academic Publishers, 1997, pp. 225-254.