Improving the Use of Multiplicity in UML Association

Hee Beng Kuan Tan, Yong Yang, Lei Bian, School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore

Abstract

Through the incorporation of derived classes and associations in class diagrams, this paper proposes a method to improve the specification power of multiplicity for n-ary associations in Unified Modeling Language (UML). The enhanced method can specify more business rules explicitly and concisely in class diagrams through the use of multiplicities instead of using constraints defined informally or in Object Constraint Language (OCL).

1 INTRODUCTION

In UML, the basic concepts for modeling structural concepts are classes and relationships [2-3]. In general, relationship types can be classified into three types: association, aggregation and generalization. Association associates different objects of associated classes. It has multiplicity. Aggregation is a more tightly coupled type of association. It models the “part-of” kind of association. Generalization is a way to structure classes to separate similarities and differences. It is to model “is-a” relationships between classes.

In an n-ary association in UML, the multiplicity at a class end specifies the number of instances of the class that are associated with each combination of instances, one from each of the remaining n-1 classes. As such, multiplicity can only be used to specify business rules that can be expressed in terms of number of instances of an associated class that are associated with each such combination. In existing method, for a business rule that actually can be expressed in terms of the number of instances of the association that associate each instance of an associated class or each combination of instances of k associated classes (1 < k < n-1), one from each class, we have to specify it using a constraint defined informally or in Object Constraint Language (OCL) [5]. For example, in the ternary association shown in Figure 1(a), the business rule, “each part is supplied by some vendors to some projects”, actually can be expressed in terms of the number of instances of the association that associate each instance of an associated class or each combination of instances of k associated classes (1 < k < n-1), one from each class, we have to specify it using a constraint defined informally or in Object Constraint Language (OCL) [5]. For example, in the ternary association shown in Figure 1(a), the business rule, “each part is supplied by some vendors to some projects”, actually can be expressed in terms of the number of instances of the association that associate each instance of an associated class or each combination of instances of k associated classes (1 < k < n-1), one from each class, we have to specify it using a constraint defined informally or in Object Constraint Language (OCL) [5].
in Figure 1(a). This problem has been discussed in [1]. This paper proposes a method to improve the specification power of multiplicity. In the proposed method, the above-mentioned business rule can be specified using multiplicity.

The paper is organized as follows. Section 2 presents the method. Section 3 concludes the paper.

2 THE PROPOSED METHOD

We propose a derived class and a derived association to improve the use of multiplicity for specifying business rules. In the use of n-ary association, if there is any business rule that can be expressed in terms of the number of instances of the association that associate each instance of an associated class or each combination of instances of k associated classes (1 < k < n-1), one from each class, we propose the inclusion of a derived class and/or a derived association that we shall discuss shortly. Through the inclusion, the business rule can be specified using multiplicity in the association included.

Let D₁, ..., Dₘ be classes (m ≥ 2). We define a derived class to represent each combination of D₁, ..., and Dₘ objects, one from each class. We shall denote such class as \( \prod_{i,j} D_i \), and call it the **product class** of D₁, ..., Dₘ. For example, let A be B two classes. We further assume that at this moment in time, the sets of A and B objects are \{a₁, a₂\} and \{b₁, b₂, b₃\} respectively. Then, the set of objects that the product class of A and B represents, is \{(a₁, b₁), (a₁, b₂), (a₁, b₃), (a₂, b₁), (a₂, b₂), (a₂, b₃)\}, where \((a_i, b_j)\) denotes the combination of \(a_i \) and \(b_j\) and \(1 ≤ i ≤ 2\) and \(1 ≤ j ≤ 3\).
Let C be an association class representing an association that associates n classes, C₁, …, Cₙ (n ≥ 1). Let B be any class from C₁, …, Cₙ or the product class of m classes from C₁, …, and Cₙ (where 1 < m < n). We define a derived association between C and B as follows:

1. If B is a class from C₁, …, Cₙ, each instance c of C is associated with the instance of B that is associated in c.
2. If B is the product class of m classes from C₁, …, Cₙ, each instance c of C is associated with the instance of B that is a combination of those Cⱼ (1 ≤ j ≤ n) instances that are associated in c.

We shall call this derived association, the **projection association** between C and B. Figure 2 gives an illustration of instances of a projection association.

Now, we shall illustrate the use of the propose method in two cases that cannot be specified by multiplicity using existing method. In the existing method, these must be specified as constraints defined informal or in OCL. Informal constraint is not as precise and explicit as multiplicity. OCL constraint is harder to understand and not so explicit in comparing with multiplicity. The first case is for the transformation of n-ary associations into binary associations. As n-ary associations (n> 2) have a rather complex structure and are not easy to understand, some propose that any n-ary association R that associates n classes, C₁, …., Cₙ, should be transformed into binary associations as follows [4]:

1. Include a class C to represent each instance of R.
2. For each j, 1 ≤ j ≤ n, include a binary association Rⱼ that associates C to Cⱼ, define the multiplicity of Rⱼ at Cⱼ end as 1.

However, in such representation, any multiplicity of R in the original n-ary association that is not “*” cannot be represented as multiplicity in these binary associations. Therefore, such multiplicity has to be defined as an informal or OCL constraint. Figure 3(a) gives an example to illustrate the problem.
With the use of the proposed method, in the above-mentioned transformation, we can represent all the multiplicities in \( R \) that are not "*" as multiplicities in binary associations in the resulting representation as follows. For each \( k, 1 \leq k \leq n \), if the multiplicity, \( m_k \), at \( C_k \) end is not "*", we include the product class \( D_k \) of \( D_1, \ldots, D_{k-1}, D_{k+1}, \ldots, \) and \( D_n \), and a projection association that associates \( D_k \) and \( C \) with multiplicity at \( D_k \) and \( C \) ends defined as 1 and \( m_k \) respectively. Figure 3(b) shows the use of the proposed method in comparing with the use of the existing method shown in Figure 3(a).

The second case is to solve the problem discussed in [1] on the use of multiplicity to specify mandatory role constraints in an n-ary association in which only \( m \) (\( 1 \leq m < n-1 \)) association roles are mandatory. With the use of the proposed method, this problem can be solved with the use multiplicity through the inclusion of the association class that represents the n-ary association, a product class of the classes that play the \( m \) association roles and a projection association that associates the association class and the product class. For example, with the use of the proposed method, the informal constraint (business rule) in Figure 1(a) can be specified as shown in Figure 1(b). It is specified as the multiplicity “1..*” at the association class (Supply) end in the binary association between Supply and Part classes.

### 3 CONCLUDING REMARK

We have proposed a novel method to improve the use of multiplicity in UML association through introducing derived class and association. With the use of the method, more business rules can be specified using multiplicity. Such specification is more concise and explicit.
Figure 3. Representing a n-ary association by binary association
REFERENCES


About the authors

Hee Beng Kuan Tan PhD in Computer Science and currently an associate professor in the School of Electrical and Electronic Engineering, Nanyang Technological University. Before moving to academic, he has 13 years of experience in software design, development and project management. His current research interests include model transformation for software development and reuse, software testing and verification, and software estimation.

Yong Yang received his B.Eng. (Hons) in Information Engineering in 2001 from Nanjing University of Science and Technology. Currently, he is a PhD student at Nanyang Technological University. His current research interests are object-oriented and component-based software design and development.

Lei Bian received her B.Sc. in Software Engineering in 2000 from Shanghai Jiao Tong University. Currently, she is pursuing a PhD degree in software engineering at Nanyang Technological University. Her current research interests include software development process and its application in the industry.