N-ary Relations of Association in Class Diagrams: Design Patterns

Sergievskiy Maxim
National Research Nuclear University MEPhI
Moscow Technological Institute
Moscow, Russia

***Abstract*—Most of the technology of object-oriented development relies on the use of UML diagrams, in particular, class diagrams. CASE tools, used for automation of object-oriented development, often do not support n-ary associations in the class diagrams, and their implementation in the form of program code in contrast to binary rather time-consuming. The article will show how in some cases it is possible to move from the n-ary association between classes to binary and how can reduce the number of objects. The rules to transform models, that contain n-ary association, will be presented in the form of design patterns. Proposed three new design patterns can be used in the process of developing software systems. These patterns describe transformations of n-ary (often ternary) associations occur between classes in binary and the introduction of additional classes and binary association with the aim of optimizing the model.**

Keywords*—UML; class diagram; multiplicity; ternary association; n-ary association; class-association; design pattern; object

I. Introduction

As it is known, UML is the standard tool for modeling software systems [1], [2], [3], [4]. Most of the object-oriented technology developments use general capabilities of this language. The design stage primarily uses class diagrams from the UML. They describe the model of a software system reflecting the main parameters of the subject area. In class diagrams, the base relationship is an association relationship. This is complex structural relation, which describes links between the objects of different classes of software system. At the later stage software system model in the form of a class diagrams will be transformed into a logical database model and object-oriented application code. It is important that a substantial part of the code can be generated automatically with the help of CASE tools. The majority of CASE tools do not support n-ary (in particular, ternary) association relationships in the class diagrams [2], [5]. Also, n-ary association, unlike binary, is a time consuming (this does not apply to databases). The article will demonstrate how in some cases it is possible to move from the n-ary association between classes (often ternary) to binary, and how you can reduce the number of potential objects of class-associations. Guidelines for the conversion of models containing n-ary association will be shown in the form of design patterns [6].

II. Replacing Ternary Association on Binary and Class-Association

Let assume that in the class diagram there is a ternary association, i.e. the association, which involves three objects. For example, take certain objects belonging to three different classes: STUDENT, SUBJECT and LECTURER (see Fig. 1). We define multiplicities for the classes from this association: STUDENT (1..*) SUBJECT (1..*), LECTURER (1).

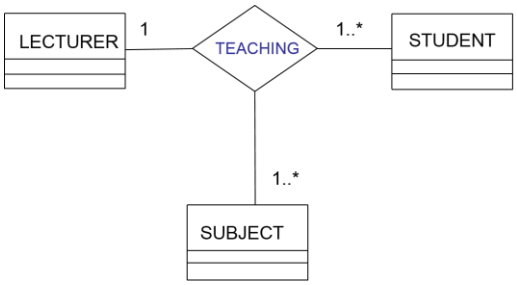

Fig. 1. The ternary association between the classes STUDENT, SUBJECT and LECTURER

The multiplicity of the association in relation to the class LECTURER is (1) because any fixed pair of objects the STUDENT and the SUBJECT corresponds to only one object class LECTURER. Each LECTURER may teach one subject with multiple students, so the multiplicity of the association in relation to the class of the STUDENT is equal to (1..*); the same lecturer can read several courses to any single student, so the multiplicity of the association in relation to the class SUBJECT equals (1..*).

Pattern_1. Assume that in the n-ary association there is a class with multiplicity (1). Then n-ary association can be replaced with a combination of (n-1)-ary association and class-association.

Proving the above is quite simple: show how it would work for a ternary association. Let's combine two classes with multiplicities, different from (1), with normal binary association. Then any two connected objects of these classes will correspond to exactly one object of the third class, which we can without loss of generality refer to the class-association. Thus, the class-association will replace the third class of ternary association. Moreover, in this class we can include attributes originally related to the ternary association.

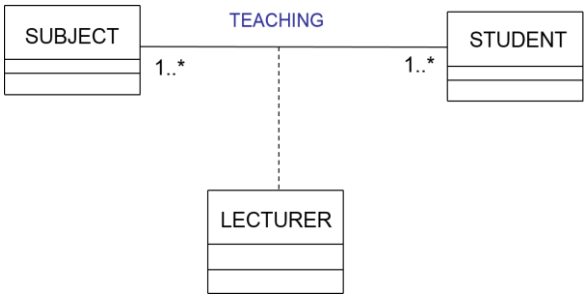

Fig. 2. Replacing ternary association on binary and class-association

Applying Pattern_1 will give us a new class diagram not containing ternary association (see Fig. 2). Such cases are fairly common. Here is another example of ternary association, which multiplicity to one of the classes (1): PLAYER (11..*), SEASON (*), CLUB (1).

The multiplicity associated with the class CLUB is equal to (1), because any player may change club only in the offseason.

The third example describes the case when all objects of the ternary associations belong to the same class - PEOPLE: this refers to the relation Father – Mother –

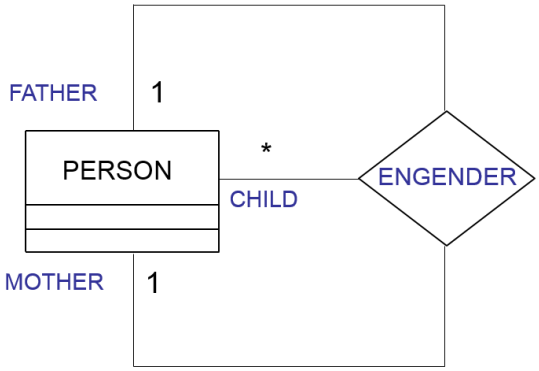

Fig. 3. Ternary association Engender (Father, Mother, Child)

Child (see Fig. 3). Here the multiplicities are as follows: PERSON (Father) - (1), PERSON (Mother) - (1), PERSON (Child) - (*).

III. Using Of Additional Classes

Let's come back to the first example. For a ternary association, as for any other, there may be relevant attributes. In this case it can be starting and finishing time of the class session and the classroom number. Let's try to strip out a subclass from a defined class of objects, which has commonalities in relation to these attributes. For example, all students are divided into different groups, for which similar classes are taking place at the same time. In this case there is a class - GROUP.

The ternary association between the classes STUDENT – SUBJECT – LECTURER is transformed into a ternary association GROUP – SUBJECT – LECTURER and the binary association STUDENT – GROUP. We define multiplicities for the classes involved in the new ternary association: GROUP (1..*), SUBJECT (1..*), LECTURER (1).

In this case, using Pattern_1, we can get the combination of simple binary association and class-association, then a class diagram will have a different appearance (see Fig. 4). The advantage of this chart is that a number of objects – instances of class-association LECTURER - will be reduced.

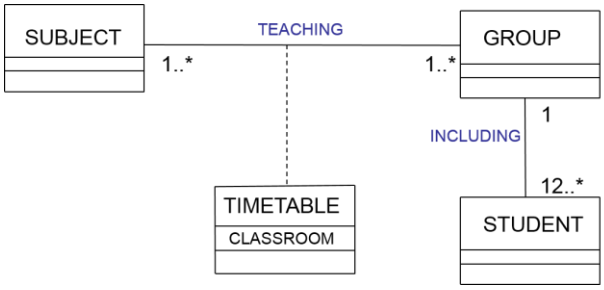

Fig. 4. The introduction of additional (to the class STUDENT) class GROUP

Based on the above, you have the following options to describe appropriate design pattern:

Pattern_2. Assume that for two or more classes there is a class-association with one or more attributes. If it is possible to split the objects of one of the classes into a subsets for which the attribute values of the class-association will be the same, then another class should be created with the association with multiplicities (1) and (*) to the first class.

But still redundancy in the form of repeated instances of a class-association LECTURER remains (in case the lecturer will teach the same subject to multiple streams). Let's introduce another class-association – TIMETABLE. This is a class, not an attribute, because it can include already specified attributes: the lecture starting time, finishing time, classroom number. This new class will be connected to a normal class-association LECTURER. New association will have the following multiplicities: (1) – for class LECTURER and (*) – for the class TIMETABLE (see Fig. 5). This solution will help to reduce the number of instances of a class-association LECTURER. Here we use an operation, similar to the operation of standard normalization from the database theory [7].

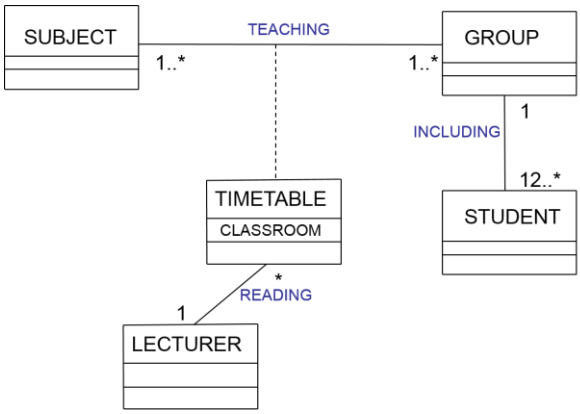

Fig. 5. Introduction of the class-association TIMETABLE

IV. Replacement of the N-Ary Association with Binary Ones

Let's describe another commonly occurring type of n-ary association that can exist between classes, or rather between objects of classes. Assume that on one side there is a single object of one class against involved in the association and on the other side there is a random number of objects of the second class. That means that the relation is defined as a set of tuples of variable length.

Let's give examples of such relations from the real subject domains [8], [9]. There are two classes: the DISEASE and GENOTYPIC_TRAIT. Relations between objects of these classes can be described as follows: object class DISEASE may be associated with any number of objects GENOTYPIC_TRAIT, and tuples of different lengths from 1 to N

(D1, GT1, ... , GTN),

characterized by an additional attribute - the probability of disease given the presence of these genotypic traits. It turns out that the tuples of the relations are of the form of:

(D1, GT1), (D1, GT1, GT2), (D2, GT2), (D2, GT3), (D2, GT1, GT2, GT3)

In case we are to show relationship between objects graphically, the result is that one object class the DISEASE may be associated with one object

GENOTYPIC_TRAIT more than once. That means the object diagram for the described relationship may be the following (see Fig. 6).

Let us go through another example. When a number of participants in a certain project are defined, the following problem often arises. The project may involve staff in different combinations. For example, performing Project_1 can attract Smit and Jones, participation could be limited to Smit only, or you can even add Clark. Resources spent in each of the above cases (time, finances, etc.) will vary, and can be added as additional attributes. Then between classes PROJECT and EMPLOYEE also encounter the recently described type associative relationship.

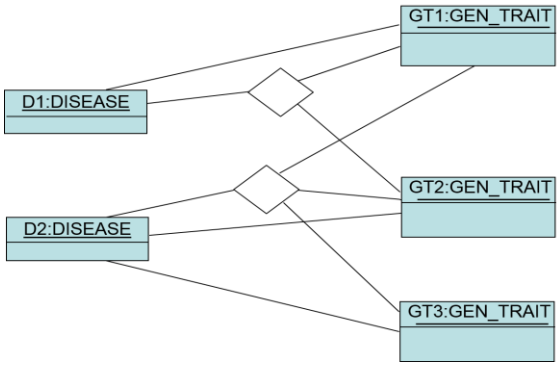

Fig. 6. Object diagram

It is obvious that means to describe and specify this relationship in UML is not. But we can solve this problem by entering additional class GROUP_GENES. In this case

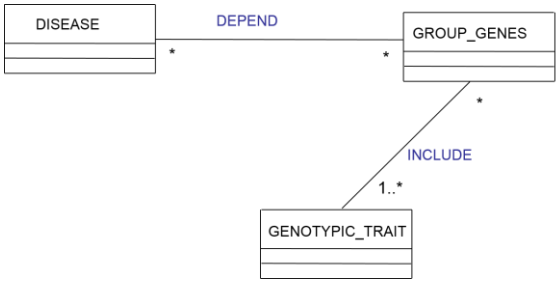

Fig. 7. Class diagram for the model genetic diseases

there is a class diagram that includes classes in addition to DISEASE and GENOTYPIC_TRAIT: the class

GROUP_GENES (see Fig. 7) and, if need, class-association between DISEASE and GROUP_GENES classes to store additional attributes.

Pattern_3. Assume that the n-ary association involves one object of the first class and random number of objects of the second class. In this case a third class should be entered to group the objects of the second class and associate it semantically different binary relations of the association with the first and second classes.

V. N-Ary Associations (N>3)

Since substantial part of this article is devoted to the n-ary association relationships, let's give examples of n-ary associations with n>3. Obviously, in real domain areas such associations are often met.

Take the domain area associated with the deliveries to the warehouses of different goods produced by different companies. We can distinguish four classes:

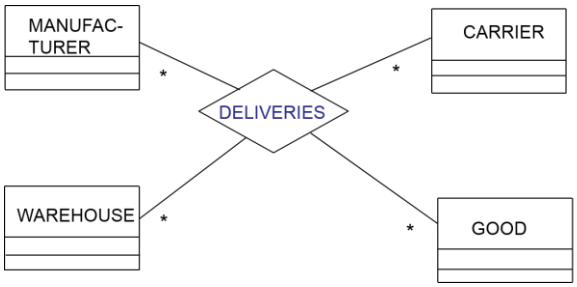

Fig. 8. Tetrary association

MANUFACTURER, WAREHOUSE, GOOD, CARRIER. Objects of these classes will be linked by relationship of association, which may have additional attributes such as delivery time, quantity, invoice number, etc. All the multiplicities in this case will be equal to (*) (see Fig. 8).

VI. Conclusion

The article describes three new design patterns which could be applied in developing software systems. These templates show transformations of n-ary (often ternary) associations, which occur between classes, into binary, as well as the introduction of additional classes and binary relations with the aim of optimizing the model. This task is very relevant given that in real domains n-ary associations are very common, and system analysts often confront with these facts

[10]. Thus, applying new templates already at the design stage make it possible: firstly to get rid of the complexity associated with modeling and realization of n-ary associations, and secondly to minimize the number objects, arising in the software system operating process.

REFERENCES

[1] G.Booch, J.Rumbaugh, I. Jacobson, "Unified Modeling Language", Addison-Wesley, 2004

[2] L.Maciaszek, "Requirements Analysis and System Design", Addison-Wesley, 2004

[3] C.Larman, "Applying UML and patterns", Prentice Hall, 2005 G. Booch,"Object-Oriented Analysis and Design with Applications", Addison-Wesley, 2007

[4] Methodical Materials IBM, https://www14.software.ibm.com

[5] E.Gamma, R.Johnson, Helm R., J.Vlissides, "Design Patterns. Elements of Reusable Object-Oriented Software", Addison-Wesley, 2001

[6] C.J. Date, "An Introduction to Database Systems", Addison-Wesley, 2004

[7] A. Konkin, M. Sergievskiy,"Integrating Bayesian Networks and Decision Trees for Calculating Probabilistic Rate of Complex Diseases", Biology and Medicine 7(3): BM-119-15-4 pages, 2015

[8] O. Lukyanchikov, E. Pluzhnik, D. Biryukov, E. Nikulchev, " Features Management and Middleware of Hybrid Cloud Infrastructures", International Journal of Advanced Computer Science and Applications, Vol. 7, No. 1, pp. 30-36, 2016

[9] G. Grenova, J. Llorens, P. Martrınez, "The meaning of multiplicity of n-ary associations in UML", Software System Modeling, No 1, pp. 86-97, 2002