On the Semantics of Redefinition, Specialization and Subsetting of Associations in UML (Extended Version)

D. COSTAL, C. GÓMEZ and P. NIETO

Abstract-- **The definition of the exact meaning of conceptual modeling concepts is considered a relevant issue since it enhances their effective and appropriate use by designers and facilitates the automatic processing of the models where they are included. Three related concepts that permit to improve the definition of an association in UML and which still lack of a formal semantic definition are: association redefinition, association specialization and association subsetting. This paper formalizes their semantics and points out the similarities and differences that exist among them. The formalization we propose is based on the meta-modelling approach and a semantic domain composed of a set of basic UML concepts and OCL expressions, which have a previous formal definition in the literature and which are well-understood.**

Index Terms-- **Association redefinition, association specialization, association subsetting, Unified Modelling Language.**

I. INTRODUCTION

DURING last decade, UML has been widely adopted both in industry and academia, thus contributing to the improvement of software engineering practices. However, one drawback of UML frequently pointed out is its lack of formal semantics. While the UML metamodel [1] gives information about its abstract syntax, its semantics is described in natural language. Thus, many concepts have not had definitions precise enough to be interpreted unambiguously.

UML 2.0 has made a significant step towards precise definitions of concepts. But its attempt to increase the expressiveness of the language has introduced new ambiguities and there are still issues that remain open.

Associations are central structural elements in UML. UML 2.0 has improved the expressiveness of the language with respect to associations in several manners, being a significant one the introduction of the association redefinition concept. This concept allows enhancing the definition of an association by means of another association that defines it more

specifically in a particular context.

Association subsetting and association specialization have been included in UML since its earliest versions and share some features with association redefinition. The three constructs involve two associations and induce an inclusion constraint between them. The similarities among the three constructs make sometimes difficult to decide which one is the best suited to model a particular situation.

Since their introduction, the three constructs have had interesting applications apart from contributing to model the relevant knowledge of an information system domain. Subsetting and redefinition of associations have been extensively used in the UML 2.0 metamodel definition [2] and in the definition of its successor UML 2.1.2 [1]. Association redefinition has shown to be of particular interest when UML is used in the process of extending the UML for a specific domain, as stated in [3] or when UML is used as a language for the definition of ontologies [4].

There has been a significant amount of work devoted to formally define the semantics of basic structural UML concepts including object classes, associations [5], [6], [7], [8], [9] and concepts related to association ends as multiplicity, aggregation and composition [10], [11], [12], [13].

In spite of these works, it seems to be generally accepted that redefinition, specialization and subsetting of associations still need to be studied in detail. As Milicev remarks [13], further research is needed to cover some important aspects related to association semantics such as association redefinition, association generalization/specialization and subsets/unions of association ends. The OMG itself in the current UML specification [1] points out that *"the interaction of association specialization with association end redefinition and subsetting is not defined"*. Rumbaugh et al. in [14] notice that *"the distinction between subsetting and*

specializing on association is not clearly described in the UML 2.0 specification". Alanen and Porres [15] mention that *"the definition of these concepts is not as straightforward as one may think and it requires an extensive study. There is an imminent need in the modeling community to standardize on one formalization of subsets and ..."*. Stevens [16] also notices that *"it is not completely clear what else subtyping for associations should really mean"*.

The main goal of this paper is to formalize the precise semantics of redefinition, specialization and subsetting of associations in UML and to make explicit the similarities and differences that exist among them. We focus on binary and non-derived associations. It has been influential the purpose of obtaining a formalization which can be understood by any designer familiar with basic UML and OCL expressions. For this reason, the formalization we propose in this paper is based on establishing a mapping between the formalized constructs and a set of basic UML concepts and simple OCL expressions, which have been formalized and which are well-understood. Some aspects of association specialization and subsetting that cannot be caught by means of a semantic formalization are described intuitively from an ontological perspective.

The precise definition of these constructs facilitates: (1) an effective and appropriate use of the constructs as well as a correct interpretation of them, thus contributing to the creation and the explanation of models that clearly communicate their intent, and (2) the machine processing of the models where they are used, namely, automatic execution, automatic reasoning and automatic generation of subsequent models as proposed by the Model Driven Architecture [17].

The rest of this paper is organized as follows. Next section reviews the basic concepts on semantic formalization and describes the approach that is used in this paper. Sections 3, 4 and 5 describe the syntax and semantics of UML association redefinition, specialization and subsetting, respectively. Section 6 compares the three constructs and points out their similarities and

differences. Section 7 reviews related work and finally, section 8 presents our conclusions and indicates some extensions to the present work.

II. Basic Concepts

There is a lot of confusion about what constitutes semantics for UML constructs. In [18] different ideas of what semantics is and wrong ways to view semantics are pointed out. Several works like [18], [19] try to clarify some of the notions involved in defining semantics for UML constructs. They distinguish the language notation of the construct (syntax) from its meaning (semantics). Next subsections describe the concepts of syntax and semantics and our approach to define the precise semantics of association specialization, redefinition and subsetting in UML.

A. Syntax

The term "syntax" is used whenever we refer to some notation. The syntax defines a language *L* of well-formed declarations and statements [19]. To define the syntax of a UML construct, both its concrete and abstract syntax must be stated.

The concrete syntax provides the rules for defining how the construct will appear when written (notation). UML, as a visual modelling language, uses lines, boxes and so on for its constructs. Its concrete syntax is described in [14], [1]. The abstract syntax identifies the main concepts onto which the concrete syntax maps [1]. In UML, the abstract syntax is provided as a model (UML metamodel). It consists of a UML class diagram with a supporting natural language description and a set of well-formedness rules written in the Object Constraint Language (OCL) [20].

B. Semantics

The semantics of a language *L* defines the meaning of each construct of the language. A semantics definition consists of two steps: first, a semantic domain must be defined and then, a mapping from the syntax to the semantic domain must be provided [19]. Therefore, the semantic

definition of a construct of a language *L* is done by mapping the construct to already known and well-understood concepts. The domain or the language where these concepts are well-understood is called semantic domain and we denote it by *S*. A mapping *M* between a language *L* and a semantic domain *S* is a function that provides for each construct of the language *L* an explanation of this construct in terms of the concepts of the semantic domain *S* (i.e., $M: L \rightarrow S$) [19].

In general, several notations or languages may be used as semantic domains. Formal languages like mathematical terms (as done in [9]), mathematical structures (as done in [21]) and Z (as done in [22]) or, even, a subset of the UML itself (as done in [6]) may be used among others. In this latter case, only a set of basic UML elements with a precise semantics definition together with OCL invariants, called UML layer, is usually considered.

In this paper we use a basic UML layer as semantic domain. The reason is that one of the main goals of this work is to make the studied constructs understandable for any UML user, even those not familiar with formal languages.

There are several approaches to formalize object-oriented modelling constructs in UML by establishing a mapping between the construct syntax and a semantic domain. The majority of these approaches have been identified by the Precise UML group in [23], [24]. One of them, used when the semantic domain is a basic UML layer, is the meta-modelling approach.

C. The Meta-modelling Approach

In this section we describe the basic steps (extracted from [25]) of the meta-modelling approach and how we apply it in this paper.

1. Develop the syntax and semantics of the core meta-modelling language selected as semantic domain. In our case this language corresponds to a basic UML layer that includes the following basic elements: classes, binary associations, multiplicities, generalization/specializations and

OCL general constraints. In the following, the semantics of each element is explained intuitively:

- *Class*. A class provides a common description for a set of elements sharing the same properties. The domain of a class is the set of objects that can be created by this class and all of its child classes (see generalization/specialization). Objects are referred to by unique object identifiers. Each object is uniquely determined by its identifier and vice versa.

- *Binary association*. Associations describe structural relationships between classes. The domain of an association is the Cartesian product of the sets of object identifiers of the participating classes. A link denoting a connection between objects is an element of that Cartesian product. We assume without loss of generality unique associations.

- *Association ends*. A binary association has two association ends, each of which connected to a class. In general, association ends may be base or derived but derived association ends are not included in our basic UML layer.

- *Multiplicity*. Multiplicity is specified for association ends. It restricts the number of links that an object can be part of.

- *Generalization/specialization*. A generalization is a taxonomic relationship between two classes (we do not consider generalization of associations in our basic layer). This relationship specializes a general class (parent class) into a more specific class (child class). Specializations and generalizations are two viewpoints of the same concept. Generalization relationships form a hierarchy over the set of classes. A generalization hierarchy induces a subset relation on the semantic domain of classes. The set of object identifiers of a child class is a subset of the object identifiers of its parent classes.

- *OCL general constraints*. A general constraint is a condition or a restriction expressed in some language for the purpose of declaring some of the semantics of an element. We assume that

they are expressed in OCL.

The concrete and abstract syntaxes of our UML layer are completely developed in [20], [1], [14] and its semantics is given using set theory as a basis and may be found in [20], [26], [9].

2. Define the abstract syntax of each UML construct. This means to transform the syntax of the UML constructs into the UML metamodel. This step is already done for the case of specialization, redefinition and subsetting of associations. Concrete and abstract syntaxes for these constructs are provided by OMG and described in detail in [1], [14]. Next sections provide an overview of both syntaxes for each construct and complement the abstract syntax defined in [1] with additional well-formedness rules when necessary.

3. Establish a mapping between the abstract syntax of each construct and the metamodel description of the semantic domain. This means to establish a mapping between the part of the metamodel that represents a construct and the basic UML layer metamodel as done, for instance, in [6]. This mapping may be established by giving directly the correspondence between both metamodels. More concretely, to provide a better understandability of these mappings, we define them as a translation between a general schema using the specified construct and another general schema using only elements defined in the basic UML layer, as done in [6].

III. UML Association Redefinition

Redefinition of associations has been incorporated to the version 2.0 of UML. It was already defined in other conceptual modelling languages such as Syntropy [27] and TAXIS [28].

A redefinition of a binary association allows us to define an association end more specifically in a particular context [1], [14].

In the following subsections an overview of the concrete and abstract syntax of UML association redefinitions is presented and a precise semantics of this construct is defined.

A. Syntax

The concrete syntax {redefines <end-name>} placed near an association end (the *redefining end*) indicates that this end redefines the one named <end-name> (the *redefined end*). Fig. 3.1 illustrates the possible scenarios of this notation. It depicts a binary association *R* with an end *b* that is redefined by a redefining end b_1. In Fig. 3.1 a) the redefining end b_1 is connected to the same class as the redefined end *b* whereas in Fig. 3.1 b) the redefining end b_1 is connected to one of the direct or indirect descendants of that class.

The binary association *R* can be recursive or non-recursive. In order to cover the recursive case, we assume that *A* and *B* may be the same class in both scenarios and A_1 and B_1 may also be the same class in scenario b).

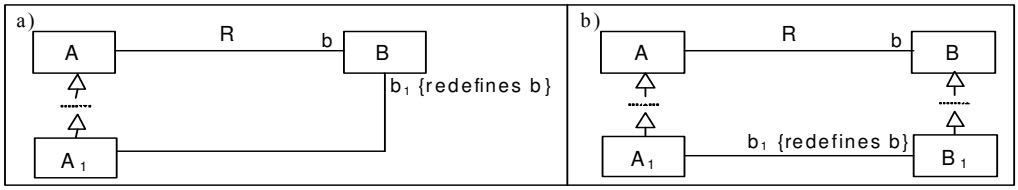


Fig. 3.1. Redefinition notation

For example, in Fig. 3.2 the end *jeProject* redefines the end *project*.

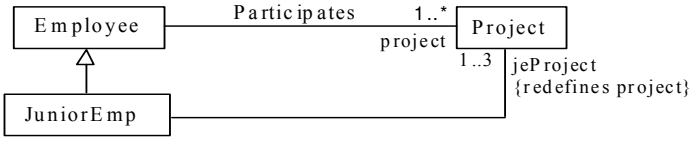


Fig. 3.2. Example of a redefinition

There is an alternative syntax for association redefinitions. An association end that has the same name as another association end that would have been inherited is assumed to redefine the inherited association end (without the need of the *redefines* keyword) [14]. Rumbaugh et al. [14] recommend not using this notation since it can easily lead to errors.

The abstract syntax of UML redefinitions is provided by the UML metamodel. Fig. 3.3 shows a fragment of the metamodel including all the concepts involved in association redefinitions.

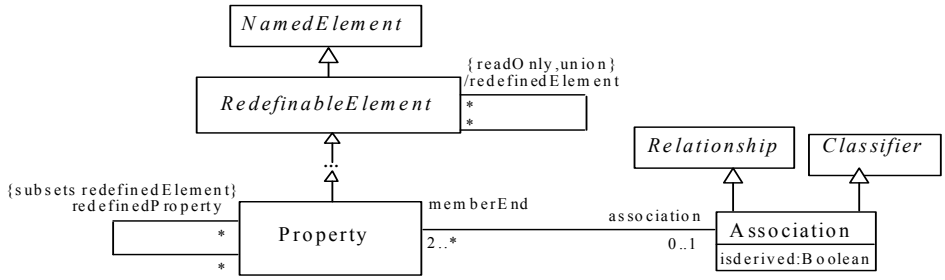


Fig. 3.3. Fragment of the UML metamodel describing association end redefinitions

In general, a redefinable element (represented in the UML metamodel as an instance of the metaclass *RedefinableElement*) is an element that, when is defined in the context of a classifier, redefines more specifically or differently another element in the context of another classifier that generalizes (directly or indirectly) the former classifier. Redefining ends of associations are instances of *Property* and their redefined ends are represented by the *redefinedProperty* role.

The redefinition of association ends is a particular case of property redefinition. In general, the characteristics of a property that can be redefined are name, type (which may be specialized), default value, derivation status, visibility, multiplicity and constraints on values.

For properties corresponding to association ends, the features that make sense to redefine are name, type, derivation status, visibility and multiplicity. In this paper, we focus on the association redefinitions that are most frequently used, name, type and multiplicity redefinitions.

The following well-formedness rules must hold for any type of association redefinition:

1. Directed and acyclical rule. Consider a graph where nodes correspond to the associations ends of a class diagram and directed arrows go from the node representing a redefining end to the node of its redefined end. The obtained directed graph must be acyclic.

2. Valid redefined property rule. A property may only redefine another property. This rule is stated graphically in the UML metamodel by means of the recursive association of *Property*.

3. Same number of ends rule. A redefining association and its redefined association must have the same number of ends.

4. Restricted end rule. The redefining end can be connected either to the same class as the redefined end or to one of its descendants.

5. Opposite end rule. The end opposite the redefining end must always be connected to a class that is a descendant of the class connected by the end opposite the redefined end.

6. Maximum multiplicity rule. The upper bound of the multiplicity specified at the redefining end, if exists, must be lower or equal to the upper bound of the multiplicity at the redefined end.

7. Minimum multiplicity rule. The lower bound of the multiplicity specified at the redefining end, if exists, must be greater or equal to the lower bound of the multiplicity at the redefined end.

Previous *valid redefined property rule*, *restricted end rule*, *opposite end rule* and *maximum* and *minimum multiplicity rules* have been established in [1]. We have complemented them with two additional necessary well-formedness rules: the *directed and acyclical rule* and the *same number of ends rule*. Notice that all of them hold in scenarios of Fig. 3.1 and in the example of Fig. 3.2.

B. Semantics

The semantic effect of a redefinition applies over a subset of the instances of the redefined association (i.e., the association with the redefined end). We call that subset the *affected instances* of the redefinition. Consider the scenarios shown in Fig. 3.1 where R denotes a redefined association that relates A and B, and A_1 denotes the class connected to the end opposite the redefining end. In all scenarios the affected instances are the instances of R which link instances of A_1 to other instances. In the Fig. 3.2 example, the affected instances are the instances of *Participates* that involve junior employees.

For any kind of association redefinition, the affected instances are the links or instances of the redefined association such that they are also links of the redefining association (i.e. they connect instances of the class at the end opposite the redefining end to another instances). There is an

implicit inclusion constraint between the redefining and redefined associations. The proof of this statement, which can be done due to our formalization, is provided in the appendix.

Next subsections describe separately the semantics of name, type and multiplicity redefinitions. After that, we present a global view of the association redefinitions semantics. Finally, last subsection shows an example where two redefinitions apply over the same association.

1) Name Redefinition

We say that a redefinition is a *name redefinition* when the redefining end has a name different from that of the redefined end. The effect of a name redefinition is to give a new name to the property at the redefined end for the affected instances of the redefinition. As a consequence, the old name of the redefined end can not be used for those instances. Name redefinitions are usually combined with the redefinition of other features in a single association redefinition although it is possible to specify a redefinition that only redefines the name. Fig. 3.4 shows a name redefinition. The name *project* is redefined by the end *jeProject*. Its effect is that the name *jeProject* will be used to refer to the projects where junior employees participate.

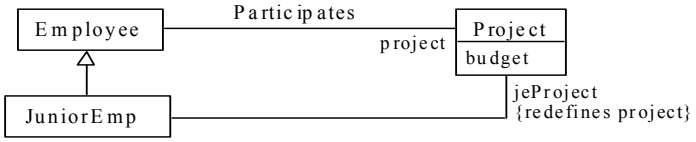


Fig. 3.4. Example of a name redefinition

To make precise the semantics of a name redefinition, we indicate in Fig. 3.5 how to translate it into our basic UML layer. The translation only depicts those elements that must be changed from the original diagram to obtain the equivalent diagram in our basic UML layer. Other elements such as attributes or the multiplicity of the redefined association are not shown since they would remain the same in the translated diagram.

The redefining association (i.e., the association with the redefining end) does not appear in the

translation. This is not surprising since the purpose of an association redefinition is not to define a new association but to improve the definition of an already existing association instead and, actually, all its instances are instances of the redefined association and can be inferred from it.

As the redefining association is not present in the translation, OCL expressions over the original diagram referring to b_1, should be rewritten to be evaluated in the translated diagram. Specifically, 'b_1' should be replaced by 'b'. In Fig. 3.5, *OCLexps* denotes a set of OCL expressions over the original diagram and *newOCLexps* denotes the set of expressions obtained by replacing 'b_1' by 'b' in *OCLexps*.

From this translation, it can be easily observed that the effect of a name redefinition is only syntactic and it has not a semantic effect on the redefined association. That is why a name redefinition is rarely used alone and it is usually combined with the redefinition of other features.

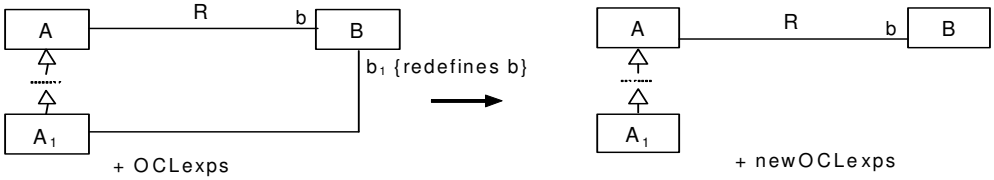


Fig. 3.5. Name redefinition translation

Consider the Fig. 3.4 example and the following OCL invariant defined over it. It forbids junior employees to participate in more than one project with a budget greater than 100000:

context JuniorEmp **inv:**
self.jeProject->select(p | p.budget>100000) -> size()<=1

Fig. 3.6 shows the translation of the diagram and its OCL expression.

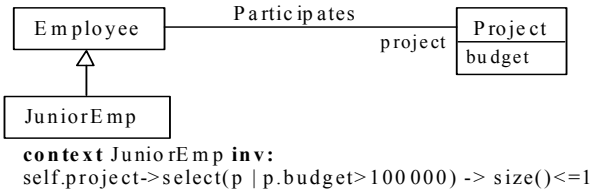


Fig. 3.6. Example of a name redefinition translation

2) Type Redefinition

In general, a redefining end can be connected either to the class at the redefined end or to one of its descendants [1]. We say that a redefinition is a *type redefinition* when the redefining end is connected to a descendant of the class at the redefined end. The effect of a type redefinition is to establish an additional *participation constraint* over the redefined association. It states that the affected instances of the redefinition must link instances of the class opposite to the redefining end to instances of the class at the redefining end. In the Fig. 3.7 example, the end *ncProject* redefines the type of the end *project* since the *NonCritical* class is a descendant of the *Project* class. The effect is that junior employees can only participate in non-critical projects. Note that the converse condition is not imposed, i.e. non-critical projects can have as participants any type of employees. The reason is that only the end *project* is redefined but not the end *employee*.

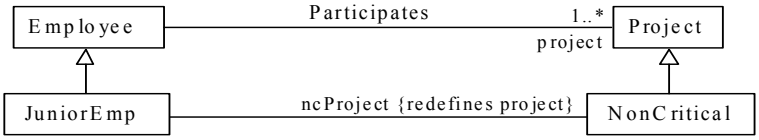


Fig. 3.7. Example of a type redefinition

The translation of a type redefinition into our basic layer consists on substituting the redefinition by a participation constraint which specifies its effect over the redefined association. Fig. 3.8 indicates the equivalence rule for this translation.

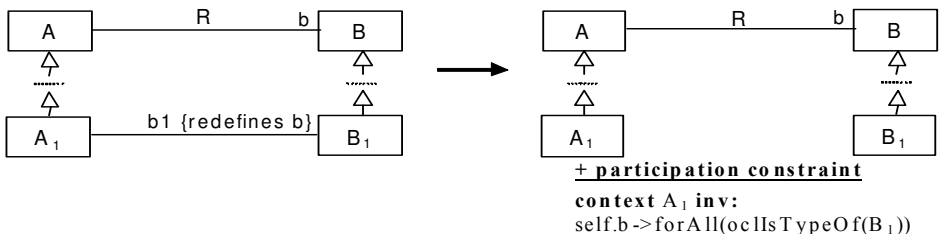


Fig. 3.8. Type redefinition translation

The participation constraint is required to ensure that each instance of *B* linked by *R* to an instance of A_1 must also be an instance of B_1. It is expressed as an OCL invariant.

Fig. 3.9 corresponds to the translation of the Fig. 3.7 example. The invariant ensures that all projects of a junior employee are non-critical projects.

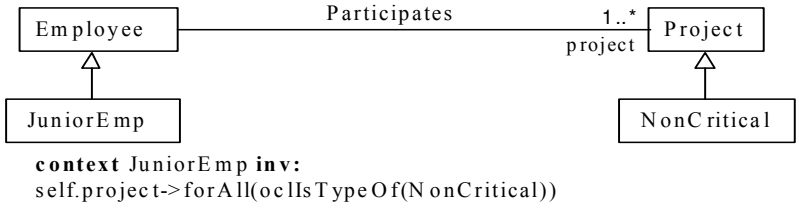


Fig. 3.9. Example of a type redefinition translation

3) Multiplicity Redefinition

We say that a redefinition is a *multiplicity redefinition* when a multiplicity is specified at the redefining end and it is more restrictive than that of the redefined end. Otherwise, multiplicity is not redefined. The effect of a multiplicity redefinition is to establish additional *cardinality constraints* over the redefined association. Basically, it restricts the multiplicity allowed for the affected instances of the redefinition. In the Fig. 3.10 example, the end *jeProject* redefines the multiplicity of the end *project*. The effect is that junior employees can not participate in more than three projects. Note that the redefining multiplicity, namely *1..3*, is more restrictive than the redefined multiplicity, *1..**.

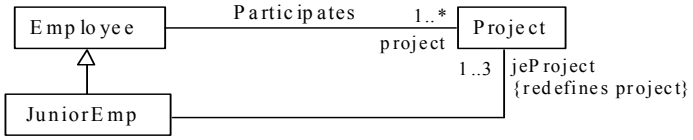


Fig. 3.10. Example of a multiplicity redefinition

Fig. 3.11 indicates how to translate a multiplicity redefinition into our basic UML layer. The redefinition is substituted by cardinality constraints which specify its effect.

The cardinality constraints ensure that each instance of the class A_1 is linked through the association *R* to a number $minb_1$ of instances at least and to a number $maxb_1$ of instances at most.

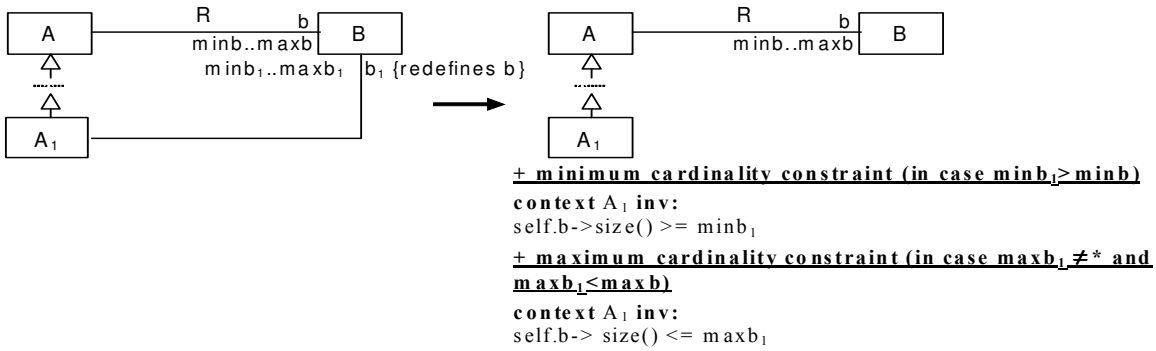


Fig. 3.11. Multiplicity redefinition translation

Fig. 3.12 shows the translation of the Fig. 3.10 example. The invariant guarantees that junior employees do not participate in more than three projects.

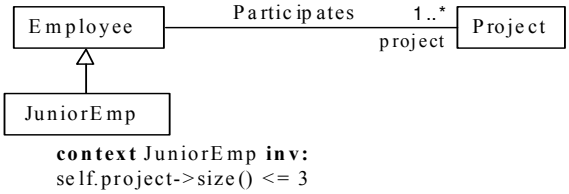


Fig. 3.12. Example of a multiplicity redefinition translation

4) Global View

Table I summarizes the results presented in previous sections. By analyzing it, it is possible to obtain a global view of the meaning of UML association redefinitions.

TABLE I: SUMMARY OF THE PROPOSED TRANSLATIONS FOR ASSOCIATION REDEFINITIONS

		Redefining association	Redefining end	Constraints
Redefined feature	Name	Not present	Substituted by the redefined end in OCL expressions	
	Type	Not present		-Participation constraint
	Multiplicity	Not present		-Minimum cardinality constraint -Maximum cardinality constraint

The main goal of an association redefinition is to improve the definition of an existing association. On the contrary, to add a new association is not the purpose of an association redefinition. This is established by the fact that the redefining association does not appear in the translation of any type of association redefinition. Moreover, the instances of the redefining association can always be inferred from those of the redefined association.

Secondly, the definition of the redefined association is improved in all cases by constraining the set of instances that are allowed for it, with the only exception of those that only redefine the

name feature. The types of constraints imposed depend on the features that are being redefined.

Name redefinitions only give a new name to the redefined property for the affected instances of the redefinition. This is why the translation of a name redefinition simply consists in replacing that name in the OCL expressions defined over the original diagram.

5) Combinations

Both ends of a binary association can be redefined. Furthermore, several redefinitions may be specified for a single association end and several features may be redefined by a single redefinition. The translation of those combinations is simply obtained by translating each of the redefined features separately and combining the result. Consider the association of Fig. 3.16 that relates people and their tickets to travel. Some people are kids as established by the specialization under class *Person*. Similarly, some tickets are kid tickets. Assume that we want to state that kids can only travel with kid tickets and kid tickets are only allowed for kids. Two redefinitions are needed, one per each end of the association. Both redefine a single feature, i.e., type.

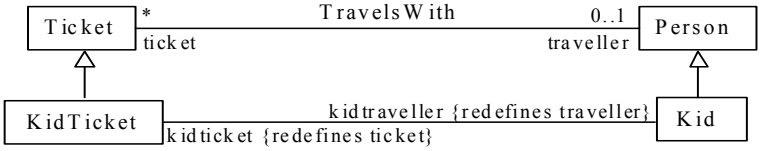
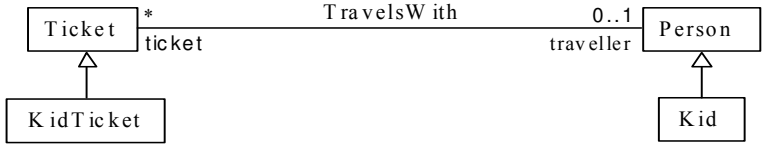


Fig. 3.16. Example of two type redefinitions for two ends of a single association

Fig. 3.17 provides the translation of the previous example.



context KidTicket **inv:**
self.traveller->forAll(oclIsTypeOf(Kid))

context Kid **inv:**
self.ticket->forAll(oclIsTypeOf(KidTicket))

Fig. 3.17. Translation of the Fig. 3.16 example

IV. UML ASSOCIATION SPECIALIZATION

The majority of conceptual modelling languages only consider the generalization/specialization

of classes. However, UML also permits the generalization/specialization of associations.

An association specialization is a taxonomic relationship between a more specific association and a more general one. Thus, the specific association inherits the features of the general association [1]. Several specializations may be specified for an association and an association may be a specialization of several associations.

A. Syntax

The concrete syntax of association specializations is shown as a line with a hollow triangle as an arrowhead between the symbols representing the involved associations. The arrowhead points to the symbol representing the general classifier. Fig. 4.1 illustrates this notation. It depicts all possible scenarios with two associations *R* and R_1 where R_1 is a specialization of *R*.

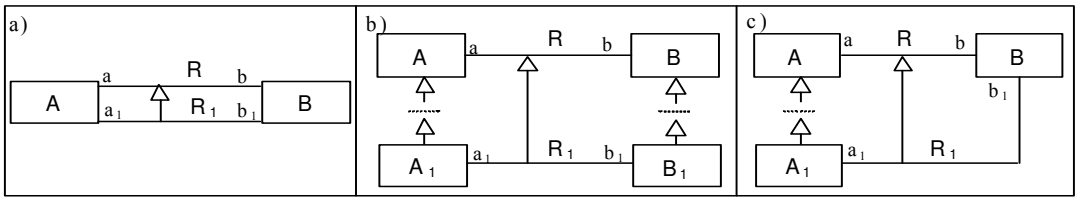


Fig. 4.1. Association generalization/specialization notation

Consider, for example, the association specialization shown in Fig. 4.2. The association *PronouncesSentence* that relates a court and a defendant is specialized by the association *Absolves*. The pronouncements of sentences in which the defendant is found not guilty by the court correspond to instances of the *Absolves* association.

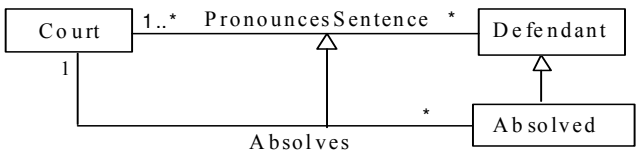


Fig. 4.2. Example of an association specialization

Fig. 4.3 shows a fragment of the UML metamodel with the concepts involved in the abstract syntax of association specialization.

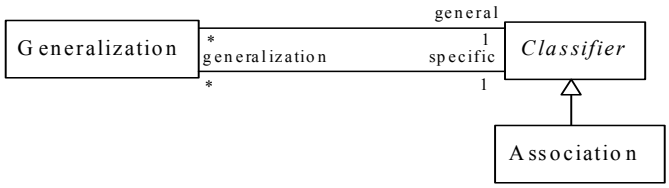


Fig. 4.3. Fragment of the UML metamodel describing association specializations

A generalization (represented in the UML metamodel as an instance of the metaclass *Generalization*) is a taxonomic relationship between a more general classifier (represented by the *general* role) and a more specific classifier (represented by the *specific* role). Associations are classifiers and, therefore, they may participate in generalizations. Specialization is a different view of the generalization relationship.

The following well-formedness rules must hold for any type of association specialization:

1. Directed and acyclical rule. Specializations hierarchies must be directed and acyclical.

2. Valid specialized classifiers rule. A classifier may only specialize classifiers of the same or a more general type. That is, associations may only specialize associations.

3. Same number of ends rule. An association that specializes another association must have the same number of ends as the other association.

4. Restricted and opposite end rule. When an association specializes another association, every end of the specific association corresponds to an end of the general association, and the specific end may be either connected to the same class as the general end or to one of its descendants.

5. Maximum multiplicity rule. The upper bound of the multiplicity of the specific association ends must be lower or equal to the upper bound of the multiplicity of their corresponding general association ends.

Rules 1, 2, 3 and 4 have been established in [1]. We have complemented them with an additional necessary well-formedness rule: the *maximum multiplicity rule*. Note that all of them hold in the Fig. 4.1 scenarios and in the Fig. 4.2 example.

B. Semantics

The effect of an association specialization is to establish an inclusion integrity constraint between the general and the specific association. This constraint ensures that each instance of the specific association must be an instance of the general association.

Then, the translation of an association specialization into our basic UML layer consists on substituting the specialization by an inclusion constraint. In OCL, constraints cannot be contextualized by an association. Therefore, the inclusion constraint is associated to one of the participant classes of the specific association. Fig. 4.4 indicates the translation for each specialization scenario defined in Fig 4.1.

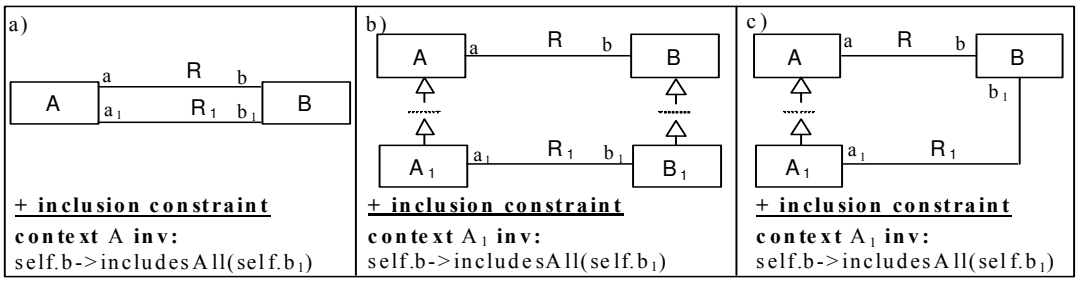


Fig. 4.4. Association generalization/specialization translation

Notice that both the general and the specific association remain in the translation. The reason is that they are two different associations representing different semantic relationships. Thus, in general and in contrast to what happened with redefinitions, the instances of the specific association cannot be inferred from those of the general association.

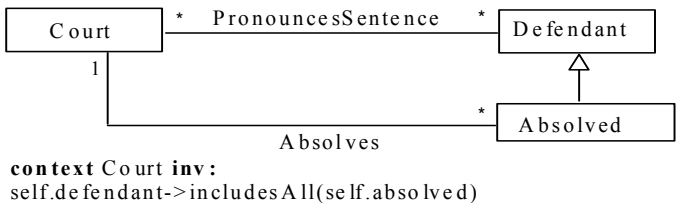


Fig. 4.5. Example of an association specialization translation

Fig. 4.5 shows the translation of the Fig. 4.2 example. The two associations remain in the

translation and the OCL invariant ensures that, for all absolutions, there exists the corresponding sentence pronouncement.

1) Ontological Perspective

The net effect of an association specialization according to the above semantic formalization is to establish an inclusion constraint between two associations. However, there is a subtle aspect, pointed out in [1], which must be considered to fully understand the meaning of the concept and to distinguish it from association subsetting (subsetting is described in the next section). [1] states that *"specialization is, in contrast to subsetting, a relationship in the domain of intentional semantics, which is to say it characterizes the criteria whereby membership in the collection is defined"*. This statement cannot be captured by a semantic formalization. Instead, an ontological analysis is required to support it.

Ontology is a branch of philosophy which deals with the order and structure of reality in the broadest sense possible and, since conceptual modelling constructs model certain types of real-world phenomena, it is possible to derive their meaning via theories of ontology [29]. There exist ontological analyses of specializations applied to classes such as [30] but not of specializations applied to associations. In this subsection, we intuitively present some notions about association specialization that further clarify their meaning. A complete formal ontological analysis is out of the scope of this paper.

First, we need the notion of *membership criterion* which has to do with the problem of recognizing instances of a certain concept. More concretely, we need the notion of *specialization membership criterion* which determines if a link or instance of a general association is an instance of a specific association or not. If we consider instances of associations as individuals, then the specialized association membership criterion can be seen as a property of those

individuals. Properties may be *intrinsic* or *extrinsic* [30]. An intrinsic property is something inherent in an individual, not dependent on other individuals. Conversely, extrinsic properties are not inherent and they have a relational nature. Therefore, the intentional semantics statement [1] about association specialization can be formulated as: the specialized association membership criterion is an intrinsic property of the instances (individuals) of the general association.

In the example of Fig. 4.2, the association *Absolves* is a specialization of *PronouncesSentence* because the criterion by which a sentence pronouncement relationship is of absolution is intrinsic to the pronouncement (a particular pronouncement is inherently of absolution or not) and it is not external to the sentence pronouncement concept. In other words, an absolution can be seen as a type of sentence pronouncement.

Consider the example of Fig. 4.6 that represents students of a faculty that define their preferences about the subjects and where students may enrol in offered subjects. Assume that the faculty establishes a policy that requires students to enrol only those offered subjects that are part of their preferences. The *Enrols* association cannot be a specialization of the *HasPreference* association since there is not a criterion, intrinsic to the *HasPreference* instances, that permits to classify them into *Enrols* instances. In fact, an enrolment is not a type of preference.

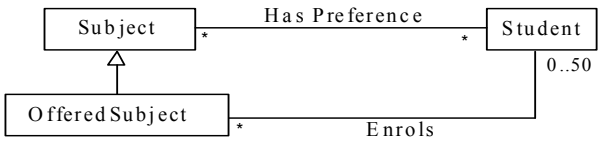


Fig. 4.6. Example of associations

A consequence of the intentional semantics of specializations is that the inclusion constraints they induce are always *analytical* constraints. Analytical constraints are a type of constraint (classified according to their source) such that its truth follows from the definition or meaning of the facts involved in it [31]. This means that an inclusion constraint associated to a specialization

comes from its own definition and, consequently, this constraint must always hold for any domain in which the specialization is defined. For instance, if we consider again the example of Fig. 4.2, we can see that the inclusion constraint between *Absolves* and *PronouncesSentence* must be always satisfied for any domain. It is not reasonable in any legal context to have the absolution of a defendant without the existence of the corresponding sentence pronouncement.

Consequently, if an inclusion constraint is fulfilled between two associations but this constraint is not analytical (it is satisfied for some domains but not for others), we can infer that there is not a specialization relationship between the associations.

Consider again the example of Fig. 4.6. The inclusion constraint between both associations is not an analytical constraint. It is easy to see that this constraint may not hold for other domains. For instance, another faculty may not take into account preferences to allow students enrolling subjects. Next section shows that association subsetting may be used to represent this constraint.

V. UML Association Subsetting

Most conceptual modelling languages, as for instance Syntropy [27] and ORM [32], permit to define subset constraints between associations.

An association subsetting in UML allows us to specify that the set of instances of an association is a subset of the set of instances of another association. An association end can be subsetted by several association ends. Furthermore, both ends of a binary association can be subsetted [1], [14]. Nevertheless, it is equivalent to specify an association subsetting between two association ends or between their respective opposite ends [15], [31].

A. Syntax

The concrete syntax {subsets <end-name>} placed near an association end (the *subsetting end*) indicates that this end subsets the one named <end-name> (the *subsetted end*). Fig. 5.1 illustrates

this notation. It depicts a binary association *R* with end *b* that is subsetted by a subsetting end b_1.

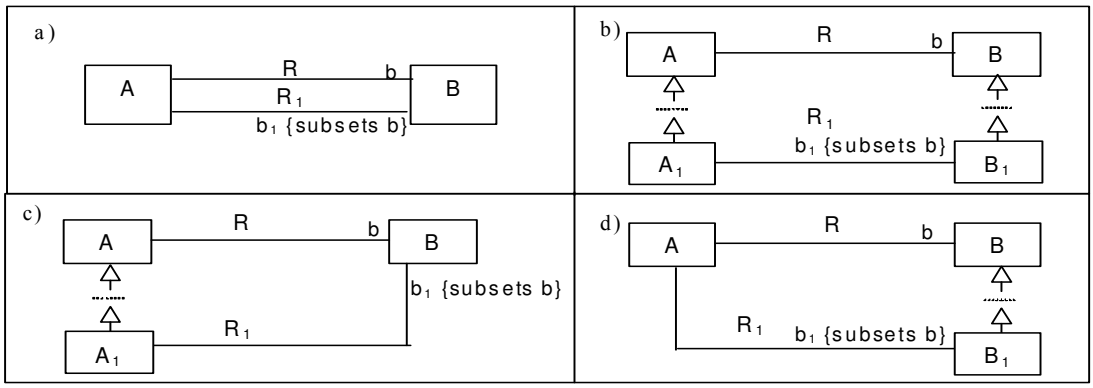


Fig. 5.1. Subsetting notation

Consider, the association subsetting shown in Fig. 5.2. The end *es* subsets the end *s*. Its effect is to ensure that all students enrolled in an offered course have a preference for that course.

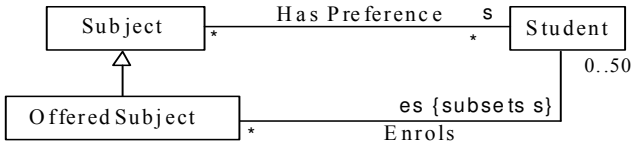


Fig. 5.2. Example of an association subsetting

Fig. 5.3 shows the fragment of the UML metamodel involved in the abstract syntax of association end subsetting.

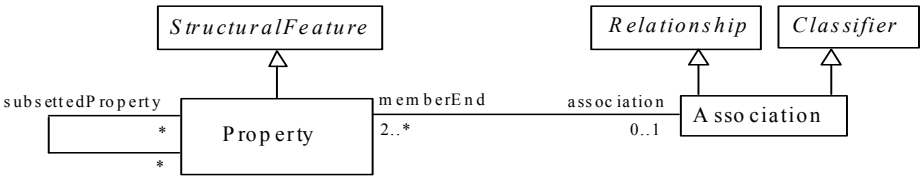


Fig. 5.3. Fragment of the UML metamodel describing association end subsetting

The subsetting of an association end is a particular case of property subsetting. It corresponds to the cases in which the subsetting property has a non-empty *association* role and, thus, it is an association end. In these cases, the *subsettedProperty* role references another association end of which it is constrained to be a subset. The following well-formedness rules must hold for association subsettings:

1. Directed and acyclical rule. Consider a graph where nodes correspond to the associations of

a class diagram and directed arrows go from the node representing a subsetting association to the node of its subsetted association of the class diagram. The obtained graph must be acyclic.

2. Valid subsetting property rule. A property may only subset another property. This rule is stated graphically in the UML metamodel by means of the recursive association of *Property*.

3. Same number of ends rule. A subsetting association and its subsetted association must have the same number of ends.

4. Restricted end rule. The subsetting end can be connected either to the same class as the subsetted end or to one of its descendants.

5. Opposite end rule. The end opposite the subsetting end can be connected either to the class at the end opposite the subsetted end or to one of its descendants.

6. Maximum multiplicity rule. If multiplicity is specified both at the subsetting and subsetted ends then the upper bound of the multiplicity at the subsetting end must be lower or equal to the upper bound of the multiplicity at the subsetted end.

7. Same end name rule. A subsetting end cannot have the same name as the subsetted end.

The rules 2, 4, 5, 6 and 7 have been established in [1]. We have complemented them with the rules 1 and 3. All these rules hold both in the Fig. 5.1 scenarios and in the Fig. 5.2 example.

B. Semantics

The effect of an association subsetting is to establish an inclusion constraint between the subsetted and the subsetting associations. Specifically, the population of the subsetting end must be included in that of the subsetted end. As it can be observed, the semantic effect of a subsetting over the instances of the involved associations is identical to that of association specializations. Therefore, the translation into our basic layer is similar to the translation of association specializations. It consists on substituting the subsetting by an inclusion constraint (see Fig. 5.5).

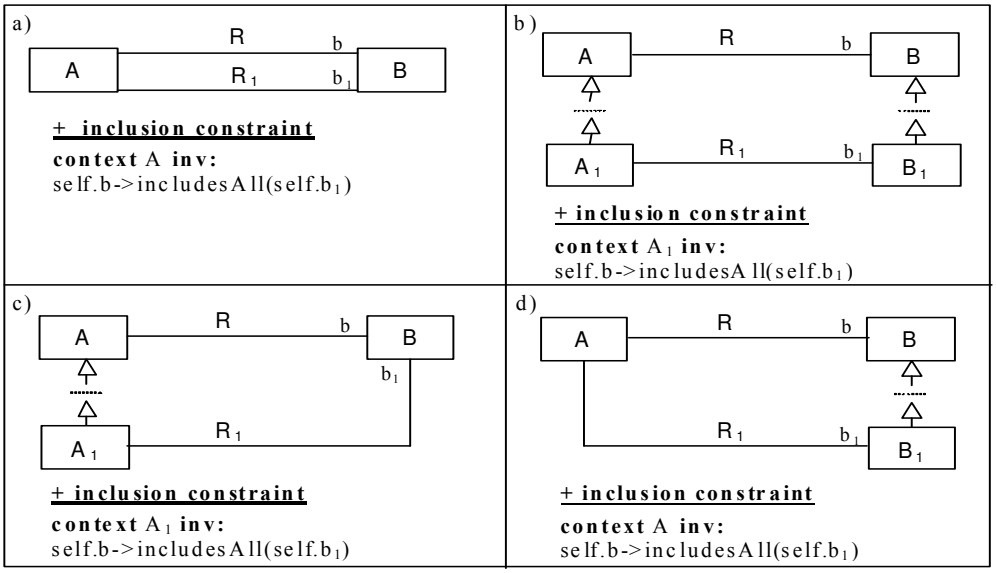


Fig. 5.5. Association subsetting translation

The inclusion constraint is required to ensure that the collection of instances for the end b_1 is a subset of the collection of instances for the end *b*. As in the association specialization case, both associations remain in the translation. The reason is that they represent different semantic relationships and, in general, the subsetting association cannot be inferred from the subsetted one. For the Fig. 5.2 example, it is not possible to infer the instances of the *Enrols* association from those of the *HasPreference* association.

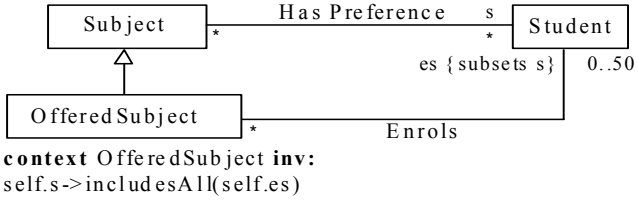


Fig. 5.6. Example of a subsetting association translation

Fig. 5.6 shows the translation of the Fig. 5.2. The invariant guarantees that the set of instances of the association *Enrols* is a subset of the set of instances of the association *HasPreference*.

1) Ontological Perspective

We have seen that a subsetting has the same semantic effect as a specialization in terms of our semantic domain. To distinguish them, we must take into account the statements in [1] about

intentional versus extensional semantics. As for association subsetting, [1] states that, in contrast to specialization, *"subsetting is a relationship in the domain of extensional semantics"* and *"subsetting represents the familiar set-theoretic concept"*. These statements cannot be captured by a semantic formalization. We must take an ontological perspective, as done for the intentional semantics of association specializations.

We intuitively present the notions needed to further clarify previous statements on subsettings and to help to distinguish them from specializations. The notion of *subsetting membership criterion* determines if a link or instance of a subsetted association is an instance of its subsetting association or not. Considering instances of associations as individuals, the subsetting membership criterion must be an *extrinsic* property of the instances (individuals) of the subsetted association. Recall that an extrinsic property is not inherent to the individual [30].

For example, there is not a criterion intrinsic to the instances of the *HasPreference* association (see Fig. 5.2) that permits to infer whether they are also instances of the *Enrols* association. The criterion by which there is an enrolment that corresponds to a particular preference is external to the particular preference.

An interesting feature related to the extensional semantics of association subsettings is that the inclusion constraints they establish may be analytical or not whereas inclusion constraints induced by specializations are always analytical.

Consider the Fig. 5.2 subsetting example, its inclusion constraint is not analytical since different faculty domains may have different policies about requiring or not the existence of a preference for allowing a student to enrol a course. By contrast, consider the subsetting of Fig. 5.7. In this case, although membership of the *Uses* association is extrinsic to the instances of the *IsAuthorizedToUse* association, the inclusion constraint between both associations is analytical

because it is reasonable to assume that, in any domain, the services used by a member of an organization must be a subset of the services he is authorized to use.

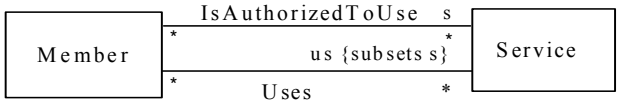


Fig. 5.7. Example of an association subsetting.

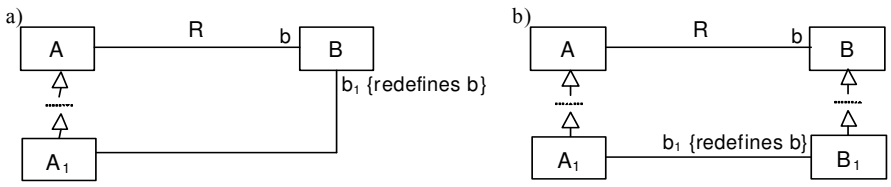
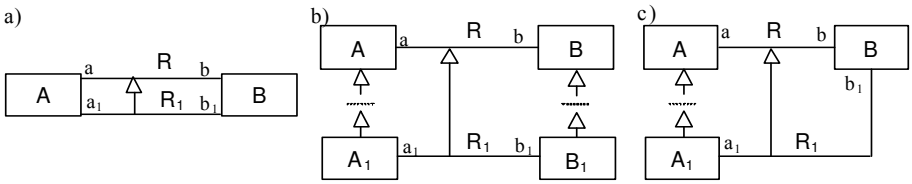
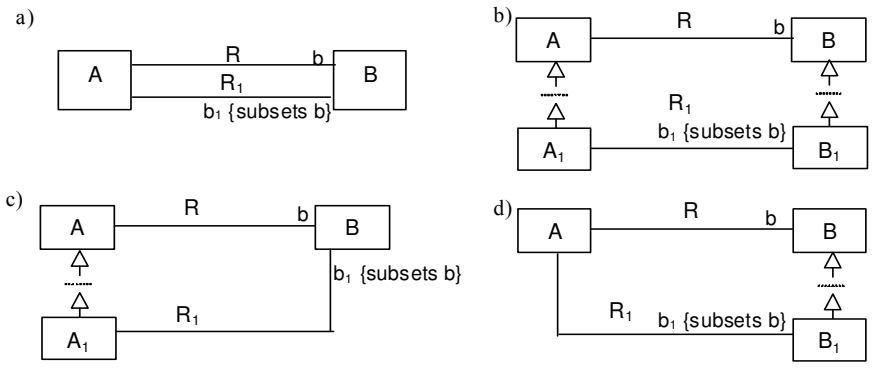
VI. Comparison among Association Redefinition, Specialization and Subsetting

In this section we describe the similarities and differences that exist among the three constructs. We focus both on their syntax and on their semantics.

A. Syntax Comparison

All scenarios of the concrete syntaxes of these constructs are shown in Table II.

TABLE II: Concrete Syntax for Association Redefinition, Specialization and Subsetting

	Concrete Syntax
Association Redefinition	
Association Specialization	
Association Subsetting	

The concrete syntaxes {redefines <end-name>} and {subsets <end-name>} of redefines and subsetting, respectively, are placed near the association end. Otherwise, the concrete syntax of association specialization consists of a line, with a hollow triangle as an arrowhead, drawn between the involved associations, and not between their ends.

The allowed scenarios for association specializations and subsettings are quite similar, as shown in Table II, whereas association redefinitions have less allowed scenarios according to their abstract syntax.

Table III summarizes the similarities and differences that exist among the abstract syntax, and concretely among the well-formedness rules of the three constructs.

TABLE III: CONCRETE SYNTAX FOR ASSOCIATION REDEFINITION, SPECIALIZATION AND SUBSETTING

		Rule							
	Applies over	**Directed and Acyclical**	**Same Number of Ends**	**Restricted End**	**Opposite End**	**Minimum Multiplicity**	**Maximum Multiplicity**	**Same End Name**	**Validness**
Association Redefinition	Association End	✓	✓	✓[1]	✓	✓	✓	×	✓
Association Specialization	Association	✓	✓	✓	✓	×	✓	×	✓
Association Subsetting	Association End	✓	✓	✓	✓	×	✓	✓	✓

[1] More restrictive

One of the aspects that differs is the model element over which the constructs apply. Specialization applies over associations whereas redefinition and subsetting apply over association ends. However, for subsettings, it is equivalent to specify a subsetting between two association ends or between their respective opposite ends while for redefinitions it is not equivalent.

The *directed and acyclical* and the *same number of ends* rules are hold for the three constructs.

The three constructs have a similar validness rule (the *valid redefined property* for

redefinitions, the *valid specialized classifier* for specializations and the *valid subsetting property* for association subsetting).

All of them have a similar *restricted end* and *opposite end rule.* However, for redefinition, the *opposite end rule* is more restrictive. It ensures that the class at the end opposite the redefining end is a descendant of the class at the end opposite the redefined end and it cannot be the same class. This is the reason why association redefinitions have less allowed scenarios in Table I.

The *maximum multiplicity rule* is stated for all the constructs whereas only association redefinitions have the *minimum multiplicity rule* defined.

A rule over the *name of the ends* is only defined for the association subsetting.

B. Semantics Comparison

The similarities and differences that exist among the semantics of association redefinition, specialization and subsetting according to the analysis described in previous sections are summarized in Table IV.

TABLE IV: SEMANTIC ASPECTS OF ASSOCIATION REDEFINITION, SPECIALIZATION AND SUBSETTING

			Constraint			
	Intrinsic Criteria	**Different Association**	**Inclusion**	**Participation**	**Minimum Cardinality**	**Maximum Cardinality**
Association Redefinition	✓ Affected instances	×	✓ Inferred Analytical	✓ Type Redef	✓ Multiplicity Redef.	✓ Multiplicity Redef.
Association Specialization	✓	✓	✓ Analytical	×	✓	×
Association Subsetting	×	✓	✓	×	✓	×

Intrinsic Criterion. The membership criterion that permits to determine whether an instance of one of the associations (i.e. the redefined, general or subsetted association) is also a member of the related association (i.e. the redefining, specific or subsetting association, respectively) may be

intrinsic or extrinsic to the instances of the first association. As Table IV indicates in its first column, this criterion must be intrinsic for association redefinition and association specialization. Conversely, it must be extrinsic for association subsetting. For the association redefinition case, the intrinsic membership criterion is fixed, that is, the instances that belong to the redefining association are necessarily the affected instances of the redefinition.

Different association. A feature that distinguishes redefinitions is that the redefining association is not an association different from that of the redefined association. The purpose of redefinitions is to improve the definition of an existing association but not to define a different one. The instances of a redefining association do not represent information different from that of the instances of its redefined association. By contrast, in specialization and subsetting, the specific and subsetting associations define associations with a meaning different from that of their general and subsetted associations, respectively.

Inclusion constraint. The three constructs establish an inclusion constraint between the involved associations. In the case of the redefinition, instances of the redefining association are also instances of the redefined association since the latter is not a new association but defines more specifically the former. Although in the specialization (subsetting) case, both associations involved have different meaning, it holds that instances of the specific (subsetting) association are instances of the general (subsetted) association. For specializations and subsettings, the inclusion constraint is explicitly stated in its translation to our basic UML layer. For redefinitions, although the inclusion constraint is not explicit in the translations, it is implied by them in all cases. The proof of this statement is provided in the appendix.

A remarkable difference related to inclusion constraints is that those established by association redefinitions and specializations are always analytical while this is not necessarily true for

association subsettings.

Participation, Maximum and Minimum Cardinality Constraints. These constraints can be induced by type and multiplicity association redefinitions, respectively. Minimum cardinality constraints are also implied by association subsettings (specializations) since the minimum cardinality of the subsetting association (specific association) is more restrictive than that of the subsetted association (general association). Conversely, association specializations and subsettings never induce participation or maximum cardinality constraints that can be specified by means of association redefinitions. The proof of these statements is provided in the appendix.

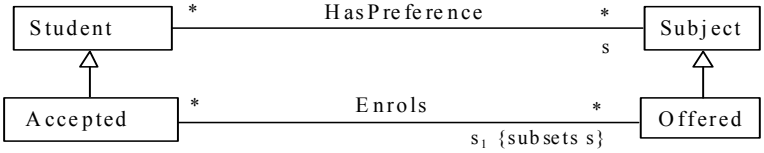


Fig. 6.1. Example of association subsetting

For example, if we consider the association subsetting depicted in Fig. 6.1, the following participation constraint is not satisfied:

context Accepted **inv:**
self.s->forAll(oclIsTypeOf(Offered))

The constraint expresses that, for an accepted student, all the subjects for which he has a preference must be offered. This constraint is not implied by the diagram, i.e. an accepted student may have a preference for a subject that is not offered.

VII. Related Work

Since associations are central structural elements in UML, there has been a considerable amount of research with the attempt to clarify and/or formalize specifically semantics of associations and the semantics of concepts related to association ends.

Génova et al. [10] deal with the interpretation of multiplicity of n-ary associations. Stevens

[16] explores some issues that are often subject to several different interpretations such as static vs dynamic associations, derived associations, multiplicity and multiple links. Barbier et al. [11] focus on the part-whole relationship concept, analyse its relation to UML aggregation and composition and propose an UML extension to fully cover it. [12] discusses a particular semantic interpretation of the UML association aggregation and composition concepts. This interpretation is obtained by applying a multidimensional framework that identifies a set of dimensions that permits to unambiguously characterize these concepts. Diskin and Dingel [33] present a framework that reconciles the structural and operational views of associations and Milicev [13] addresses uniqueness and ordering of association ends. Some of these woks provide a formalization to describe the semantics of the studied issues [11], [33], [13]. Others [12], [10], [16] provide precise descriptions that clarify them.

Association redefinition is not a new concept in conceptual modeling. It was already considered in conceptual modelling languages as Syntropy [27] and TAXIS [28]. Syntropy permits to define type and multiplicity association redefinitions but not name redefinitions. TAXIS only considers type redefinitions.

Some authors have studied the concept of association redefinition in UML. Kleppe and Rensink [21], [34] use type graphs to represent UML class diagrams. Classes are represented as nodes, attributes and associations are represented as unidirectional edges with their name as label and generalization/specialization relationships between classes are represented as unlabelled arrows with a triangular arrow head. Association redefinitions (only type redefinitions are considered) are specialization constraints between edges. These constraints are precisely defined as a formal extension of type graphs indicating that the edge corresponding to a redefined end is overridden by the edge corresponding to its redefining end. Büttner and Gogolla [35] describe the

concept of redefinition applied to methods, attributes and associations and discusses which consequences arise when this construct is implemented. Moreover, they point out some concerns with the UML 2.0 specification of this construct. In a previous work [36], we describe how to use association redefinitions and analyze the interactions between taxonomic constraints and association redefinitions. Moreover, we establish some conditions that are necessary to guarantee well-formed association redefinitions. Olivé [31] uses UML association redefinitions to describe participant refinements and cardinality constraint strengthening [37].

Very little works have dealt with the association specialization concept in UML. This construct has been analysed by Stevens in [16]. This work states that specialization for associations in UML 1.4 may be used in two different contexts: to represent an inclusion constraint between associations and to represent a participation constraint over an association (i.e. a type redefinition). Olivé [31] defines that a relationship type R' is a specialization of a relationship type R if R' has the defining properties of R and others. R' is more specialized because it contains more defining properties than R does.

Most conceptual modelling languages, as for instance Syntropy [27] and ORM [32], permit to define subset constraints between associations as UML does. A lot of works have dealt with the association subsetting concept in UML. Kleppe and Rensink [34] formalize this concept as a formal extension of a type graph. Concretely, the formalization states that an edge of a graph can be declared as subset of another if both its source node and its target node specialize those of the other. This means that only one of the possible scenarios for this construct is considered (see Fig 5.1, scenario b)). The meaning for this construct is that an edge of the superclass should exist whenever an edge of the subclass exists in the instance graph. Alanen and Porres [15] present a set-theoretic formalization of metamodels and models and define pre- and postconditions for

basic operations on element creation, deletion and modification, including the subset property. According to Olivé [31], association subsetting represents an inclusion constraint that cannot be seen as an association specialization. Moreover, the work points out by means of examples the differences between the association specialization and subsetting.

The work presented here contributes to the related work of the area in several aspects. We give a precise semantics of UML association redefinition whereas the majority of mentioned previous works only describe the construct and its use. Just [21] provides a formalization but only for type redefinitions. In the same way, we provide an unambiguous semantics for association specialization. The semantics for this construct has not been defined in the literature. Only [31] provides a clear definition. As far as UML association subsetting concerns, we provide its formal semantics as [34], [15] do and we give some hints to distinguish this construct from specialization of associations. Moreover, we have analyzed the abstract syntax of these constructs and we have added some missing well-formedness rules. Finally, we have compared the three constructs remarking the similarities and differences.

VIII. Conclusions and Future Work

This paper has studied three constructs related to UML associations. Concretely, the paper reviews the syntax and defines the precise semantics of the association redefinition, association specialization and association subsetting. These constructs are not clearly defined in the UML 2.0 specification and their semantics is, in some aspects, ambiguous.

We have stated the semantics for these constructs by establishing a mapping between their abstract syntax and a set of already known and well-understood UML concepts that we called basic UML layer. These mappings provide a precise explanation of these constructs by means of the concepts clearly defined in the UML layer. Moreover, the paper has analyzed the similarities

and differences concerning the syntax and the semantics among the three constructs. This analysis may help designers to choose the appropriate construct to model a specific domain.

As a further research, we would like to extend this work in several directions. We are interested in defining the semantics of these three constructs considering derived associations as well as other types of association redefinitions. We are also interested in studying other constructs related to UML associations as union properties. Moreover, we want to complete and formalize the ontological analysis presented in this paper. Additionally, we plan to define a set of analysis patterns to help designers to select the more suitable construct for a concrete domain.

APPENDIX

Theorem 1: *Participation constraint not entailed by association subsetting (specialization)*
Let *A* and *B* be two classes and *R* a binary association between them. Let *b* be the association end that connects *R* to class *B*. Let A_1 be a subclass of *A*, B_1 a subclass of *B* and R_1 a binary association between them. Let b_1 be the end that connects R_1 to B_1. Assume that the end b_1 subsets the end *b* (R_1 is a specialization of *R*). Then, the participation constraint expressed as **context** A_1 **inv:** `self.b->forAll(oclIsTypeOf(B1))` is not entailed by the association subsetting (specialization) (see Fig. A.1).

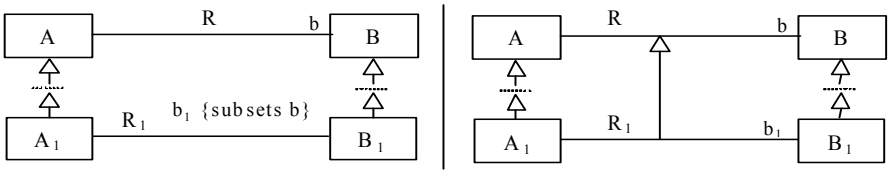
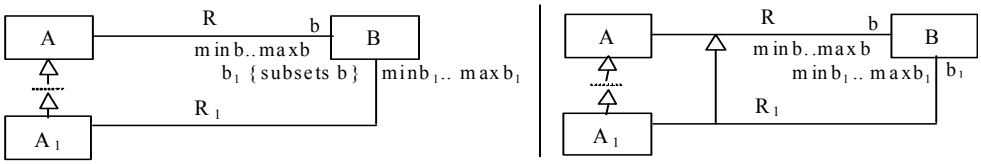


Fig. A.1. Participation constraint not entailed by subsetting or specialization

Proof. (1) The translation of the declared subsetting (specialization) into our basic UML layer requires the inclusion constraint expressed as **context** A_1 **inv:** `self.b -> includesAll(self.b1)`. Fig. 5.5 b) (Fig. 4.4 b)) depicts the translated schema.
(2) Assume an information base (IB) where:
(2.1) *x* and *y* are instances of *B* and *x* is also an instance of B_1.
(2.2) *z* is an instance of *A* and A_1.
(2.3) *z* is *R-related* to *x* and to *y* and *z* is R_1*-related* to *x*.
(3) The IB described in (2) satisfies all the constraints of the translated schema described in (1), i.e. it satisfies the inclusion constraint together with the graphical constraints stated over the translated diagram.
(4) The IB described in (2) does not satisfy the participation constraint because, according to that constraint, instances *x* and *y* should be of type B_1, and, from step (2) only *x* is an instance of B_1.
(5) Therefore, from (3) and (4) the participation constraint is not entailed by the association subsetting (specialization).

Theorem 2: *Minimum cardinality constraint entailed by association subsetting (specialization)*

Let *A* and *B* be two classes and *R* a binary association between them. Let *b* be the association end that connects *R* to class *B*. Let A_1 be a subclass of *A* and R_1 a binary association between A_1 and *B*. Let b_1 be the end that connects R_1 to *B* and $minb_1$ the minimum cardinality at the end b_1. Assume that the end b_1 subsets the end *b* (R_1 is a specialization of *R*). Then, the minimum cardinality constraint expressed as **context** A_1 **inv:** `self.b->size() >= minb1` is entailed by the association subsetting (specialization) (see Fig. A.2).

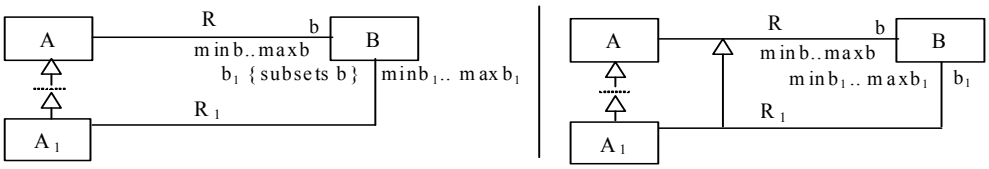


(entailed) **context** A_1 **inv**: self.b->size() >= $minb_1$

Fig. A.2. Minimum cardinality constraint entailed by subsetting or specialization

Proof. (1) The translation of the declared association subsetting (specialization) into our basic UML layer requires the inclusion constraint expressed as **context** A_1 **inv:** `self.b -> includesAll(self.b1)`. Fig. 5.5 c) (Fig. 4.4 c)) depicts the translated schema.
(2) Since *self.b* contains all the elements of $self.b_1$ the size of *self.b* must be greater than or equal to the size of $self.b_1$.Then, self.b->size() >=self.b_1->size().
(3) Since $minb_1$ is the minimum cardinality at the end b_1 the size of $self.b_1$ must be greater than or equal to $minb_1$. Then, self.b_1->size() >= $minb_1$.
(4) Therefore, from (2) and (3) the minimum cardinality constraint is entailed by the association subsetting (specialization).

Theorem 3: *Maximum cardinality constraint not entailed by association subsetting (specialization)*
Let *A* and *B* be two classes and *R* a binary association between them. Let *b* be the association end that connects *R* to class *B* and *maxb* the maximum cardinality at the end *b*. Let A_1 be a subclass of *A* and R_1 a binary association between A_1 and *B*. Let b_1 be the end that connects R_1 to *B* and $maxb_1$ the maximum cardinality at the end b_1. Assume that the end b_1 subsets the end *b* (R_1 is a specialization of *R*). Then, the maximum cardinality constraint expressed as **context** A_1 **inv:** `self.b->size() <= maxb1` is not entailed by the association subsetting (specialization) (see Fig. A.3).



(not entailed) **context** A_1 **inv**: self.b->size() <= $maxb_1$

Fig. A.3. Maximum cardinality not entailed by *subsetting or specialization*

Proof. (1) The translation of the declared subsetting (specialization) into our basic UML layer requires the inclusion constraint expressed as **context** A_1 **inv:** `self.b -> includesAll(self.b1)`. Fig. 5.5 c) (Fig. 4.4 c)) depicts the translated schema.
(2) Assume an information base (IB) where:
(2.1) *x* and *y* are instances of *B*.
(2.2) *z* is an instance of *A* and A_1.
(2.3) maxb = 2 and $maxb_1 = 1$.
(2.4) *z* is *R-related* to *x* and to *y* and *z* is R_1*-related* to *x*.
(3) The IB described in (2) satisfies all the constraints of the translated schema described in (1), i.e. it satisfies the inclusion constraint together with the graphical constraints stated over the translated diagram.
(4) The IB described in (2) does not satisfy the maximum cardinality constraint because, according to that constraint, *z* can be *R-related* at most to an instance of type *B*, and, from step (2) *z* is *R-related* to *x* and to *y*.

(5) Therefore, from (3) and (4) the maximum cardinality constraint is not entailed by the association subsetting (specialization).

Theorem 4: *Inclusion constraint is entailed by any type of redefinition (i.e. name, type and multiplicity)*
Let *A* and *B* be two classes, *R* a binary association between them and A_1 a subclass of *A*. Let *b* be the association end that connects *R* to class *B*. Assume that b_1 is a redefinition of the end *b* and that its opposite end is connected to A_1. Then, the inclusion constraint expressed as **context** A_1 **inv:** `self.b->includesAll(self.`b_1`)` is entailed by the association redefinition (see Fig. A.4).

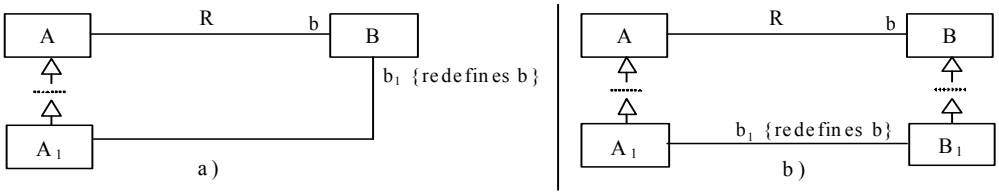


Fig. A.4. Inclusión constraint entailed by any type of redefinition

Proof. (1) We distinguish two cases:
(1.1) The end b_1 also redefines the name of the end *b*. Then, the translation of the redefinition into our basic UML layer replaces b_1 by *b* in OCL expressions and the inclusion constraint is translated to **context** A_1 **inv:** `self.b->includesAll(self.b)`
(1.2) The end b_1 redefines the end b but both end names are equal. Then, the inclusion constraint can also be formulated as **context** A_1 **inv:** `self.b->includesAll(self.b).`
(2) Since *self.b* contains all the elements of *self.b*, the inclusion constraint is entailed by the redefinition in both cases.

REFERENCES

[1] Object Management Group: Unified Modeling Language (OMG UML), Superstructure, V2.1.2, OMG Available Specification without Change Bars (formal/2007-11-02). (2007).
[2] Object Management Group: Unified Modeling Language 2.0 Superstructure Specification, OMG Adopted Specification, 2005. Available online at http://www.omg.org/cgi-bin/doc?formal/05-07-04.
[3] Burgués, X., Franch, X., Ribó, J.M.: Improving the accuracy of UML metamodel extensions by introducing induced associations. Software and Systems Modeling 7 (2008) 361-379.
[4] Conesa, J., Olivé, A.: A Method for Pruning Ontologies in the Development of Conceptual Schemas of Information Systems. J. Data Semantics V: 64-90 (2006)
[5] France, R. B.: A Problem-Oriented Analysis of Basic UML Static Requirements Modeling Concepts. In Proc. ACM SIGPLAN Conf. Object-Oriented Programming, Systems, Languages and Applications, (1999), pp. 57-69.
[6] Gogolla, M., Richters, M.: Expressing UML Class Diagrams Properties with OCL, in: Object Modeling with the OCL, LNCS, vol. 2263 (2002), pp. 85-114.
[7] Kuske, S., Gogolla, M., Kollmann, R., Kreowski, H-J.: An Integrated Semantics for UML Class, Object and State Diagrams Based on Graph Transformation. In: Integrated Formal Methods, LNCS, vol. 2335, (2002) pp. 11-28.
[8] Richters, M., Gogolla, M.: On Formalizing the UML Object Constraint Language OCL, In: International Conference on Conceptual Modeling , LNCS, vol. 1507 (1998) pp. 449 - 464.
[9] Szlenk, M.: Formal Semantics and Reasoning about UML Class Diagram, In: Proceedings of the International Conference on Dependability of Computer Systems, IEEE Computer Society, (2004), pp. 51-59.
[10] Génova, G., Llorens, J., Martínez, P.: The meaning of multiplicity of n-ary associations in UML. Software and Systems Modeling 1 (2002) 86-97.
[11] Barbier, F., Henderson-Sellers, B., Le Parc-Lacayrelle, A., Bruel, J. M.: Formalization of the Whole-Part Relationship in the Unified Modelling Language. IEEE Transactions on Software Engineering 29 (2003) 459-470.
[12] Albert, M., Pelechano, V., Fons, J. et al.: Implementing UML Association, Aggregation, and Composition. A Particular Interpretation Based on a Multidimensional Framework. In: Advanced Information Systems Engineering , LNCS, vol. 2681, (2003), pp. 143-158.
[13] Milicev, D.: On the Semantics of Associations and Association Ends in UML. IEEE Transactions on Software Engineering 33 (2007) pp. 238-251.
[14] Rumbaugh, J., Jacobson, I., Booch, G.: The unified modeling language reference manual, 2nd edn, (Addison-Wesley, 2005).
[15] Alanen, M., Porres, I.: Basic Operations over Models Containing Subset and Union Properties. In Nierstrasz, O., Whittle, J., Harel, D., Reggio, G. (Eds.), Model Driven Engineering Languages and Systems, LNCS, vol. 4199, (2006), pp. 469-483.

[16] Stevens, P.: On the interpretation of binary associations in the Unified Modelling Language. Software and Systems Modeling 1 (2002), pp. 68-79.
[17] Object Management Group: MDA Guide Version 1.0.1, OMG, omg/2003-06-01, 2003.
[18] Harel, D., Rumpe, B.: Meaningful Modeling: What's the Semantics of "Semantics"?., In: IEEE Computer, vol 37, n. 10, (2004), pp. 64-72.
[19] Rumpe, B.: A Note on Semantics (with an Emphasis on UML), In: European Conference on Object-Oriented Programming Workshops, Workshop on Precise Behavioral Semantics (with an Emphasis on OO Business Specifications), LNCS, vol. 1543 (1998).
[20] Object Management Group: Object Constraint Language (OCL), Version 2.0, OMG Available Specification (formal/2006-05-01). (2006) http://www.omg.org/docs/ptc/03-10-14.pdf.
[21] Kleppe, A., Rensink, A.: On a Graph-Base Semantics for UML Class and Object Diagrams, In: Electronic Communications of the EASST,vol. 10, (2008)
[22] Zhan, X., Miao, H.: An Approach to Formalizing the Semantics of UML Statecharts, in: International Conference on Conceptual Modeling, LNCS, vol. 3288 (2004) pp. 753-765.
[23] Andreopoulos, W.: Defining Formal Semantics for the Unified Modeling Language, Research Report of Department of Computer Science, University of Toronto, (2000) http://www.cs.toronto.edu/~chechik/courses99/csc2108/projects/index.html.
[24] France, R., Evans, A., Lano, A., Rumpe, B.: The UML as a Formal Notation, In: The Unified Modeling Language (UML'98) - Beyond the Notation. First International Workshop, LNCS, vol. 1618, (1998) pp. 336-348.
[25] Kent, S., Evans, A., Rumpe, B.:UML Semantics FAQ, In: European Conference on Object-Oriented Programming Workshops, LNCS, vol. 1743, (1999) pp. 33-56.
[26] Richters, M., Gogolla, M.: On Formalizing the UML Object Constraint Language OCL, In: International Conference on Conceptual Modeling, LNCS, vol. 1507 (1998) pp. 449 - 464.
[27] Cook, S., Daniels, J.: Designing Object Systems: Object-Oriented Modelling with Syntropy, (Prentice Hall, 1994).
[28] Milopoulos, J., Bernstein, P.A., Wong, H.K.T.: A Language facility for Designing Database-Intensive Applications. TODS, vol. 5, n. 2 (1980) pp. 185-207.
[29] Wand, Y., Storey, V.C., Weber, R.: An Ontological Analysis of the Relationship Construct in Conceptual Modeling. ACM Trans. Database Syst. 24(4), (1999), pp. 494-528.
[30] Welty, C.A., Guarino, N.: Supporting ontological analysis of taxonomic relationships. Data Knowl. Eng. 39(1): 51-74 (2001).
[31] Olivé, A.: Conceptual modeling of information systems, (Springer-Verlag, 2007).
[32] Halpin, T.: Information Modeling and Realtional Databases: From Conceptual Analysis to Logical Design, (Morgan Kaufmann, 2001)
[33] Diskin, Z., Dingel, J.: Mappings, Maps and Tables: Towards Formal Semantics for Associations in UML2. In Model Driven Engineering Languages and Systems, LNCS, vol. 4199, (2006), pp. 230-244.
[34] Kleppe, A., Rensink, A.: A Graph-Based Semantics for UML Class and Object Diagrams, in: Technical Report TR-CTIT-08-06 Centre for Telematics and Information Technology, University of Twente, Enschede, (2008)
[35] Büttner, F., Gogolla, M.; On Generalization and Overriding in UML 2.0, UML Workshop on OCL and Model Driven Engineering, Lisbon (Portugal), 2004.
[36] Costal, D., Gómez, C.: On the Use of Association Redefinition in UML Class Diagrams. In: Conceptual Modeling, LNCS, vol. 4215, (2006), pp. 513-527.
[37] Costal, D., Olivé, A., Teniente, E.; Relationship Type Refinement in Conceptual Models with Multiple Classification. In: 20th International Conference on Conceptual Modeling., LNCS, vol. 2224, pp. 397-411.